

# Manual de Live Systems

Projecte Live Systems <[debian-live@lists.debian.org](mailto:debian-live@lists.debian.org)>

---

Copyright © 2006-2014 Live Systems Project

Aquest programa és un programari lliure: es pot redistribuir i/o modificar sota els termes de la Llicència Pública General de la GNU com és publicada per la Free Software Foundation, ja sigui la versió 3 de la Llicència, o (si ho preferiu) qualsevol versió posterior.

Aquest programa es distribueix amb l'esperança que sigui útil, però sense cap garantia, fins i tot sense la garantia implícita de COMERCIALITZACIÓ o ADEQUACIÓ PER A PROPÒSITS DETERMINATS. Vegeu la Llicència General Pública de la GNU per a més detalls.

Haurieu de rebre una còpia de la Llicència Pública General de la GNU amb aquest programa. Si no és així, consulteu <http://www.gnu.org/licenses/>.

El text complet de la Llicència Pública General de la GNU es pot trobar a /usr/share/common-licenses/GPL-3.

## Contents

<b>Sobre aquest manual</b>	<b>2</b>
<b>Sobre aquest manual</b>	<b>3</b>
<b>1. Sobre aquest manual</b>	<b>3</b>
1.1 Per als impacients	3
1.2 Termes	3
1.3 Autors	4
1.4 Contribuir a aquest document	5
1.4.1 Aplicar canvis	5
1.4.2 Traducció	6
<b>Sobre el Live Systems Project</b>	<b>7</b>
<b>2. Sobre el Live Systems Project</b>	<b>7</b>
2.1 Motivació	7
2.1.1 Què passa amb els sistemes vius actuals	7
2.1.2 Per què crear el nostre pròpi sistema viu?	7
2.2 Filosofia	7
2.2.1 Només paquets Debian sense modificacions de la secció “main”	7
2.2.2 Paquets del sistema viu sense cap configuració	8
2.3 Contacte	8
<b>Usuari</b>	<b>9</b>

<b>Instal·lació</b>	<b>10</b>
<b>3. Instal·lació</b>	<b>10</b>
3.1 Requisits	10
3.2 Instal·lació de live-build	10
3.2.1 Des del repositori de Debian	10
3.2.2 A partir del codi font	10
3.2.3 A partir d'instantànies	11
3.3 Instal·lació de live-boot i live-config	11
3.3.1 Des del repositori de Debian	11
3.3.2 A partir del codi font	11
3.3.3 A partir d'instantànies	12
<b>Conceptes bàsics</b>	<b>13</b>
<b>4. Conceptes bàsics</b>	<b>13</b>
4.1 Què és un sistema viu?	13
4.2 Descàrrega d'imatges prefabricades	14
4.3 Ús del servei de construcció d'imatges en viu web	14
4.3.1 Ús i advertències sobre el servei web	14
4.4 Primers passos: construcció d'una imatge ISO híbrida	15
4.5 Usar una imatge ISO híbrida en viu	15
4.5.1 Gravar una imatge ISO en un medi físic	15
4.5.2 Copiar una imatge ISO híbrida en un dispositiu USB	16
4.5.3 Utilitzar l'espai lliure en una memòria USB	16
4.5.4 Arrencar el medi en viu	16
4.6 Utilitzar una màquina virtual per a fer proves	17
4.6.1 Provar una imatge ISO amb QEMU	17
4.6.2 Provar una imatge ISO amb VirtualBox	17
4.7 Construir i utilitzar una imatge HDD	18
4.8 Construir una imatge netboot	19
4.8.1 Servidor DHCP	20

4.8.2 Servidor TFTP	20
4.8.3 Servidor NFS	20
4.8.4 Com provar l'arrencada en xarxa	21
4.8.5 Qemu	21
4.9 Webbooting	21
4.9.1 Obtenir els fitxers webboot	21
4.9.2 Arrencar imatges webboot	22
<b>Descripció general de les eines</b>	<b>23</b>
<b>5. Descripció general de les eines</b>	<b>23</b>
5.1 El paquet live-build	23
5.1.1 L'ordre lb config	23
5.1.2 L'ordre lb build	24
5.1.3 L'ordre lb clean	24
5.2 El paquet live-boot	24
5.3 El paquet live-config	24
<b>Gestió d'una configuració</b>	<b>25</b>
<b>6. Gestió d'una configuració</b>	<b>25</b>
6.1 Gestionar canvis en la configuració	25
6.1.1 Per què utilitzar scripts auto? Què fan?	25
6.1.2 Utilitzar scripts auto d'exemple	25
6.2 Clonar una configuració publicada via Git	26
<b>Personalització dels continguts</b>	<b>27</b>
<b>7. Visió general de la personalització</b>	<b>27</b>
7.1 Configuració durant la construcció vs. durant l'arrencada	27

7.2 Etapes de la construcció	27
7.3 Suplementar lb config amb fitxers	28
7.4 Tasques de personalització	28
<b>Personalització de la instal·lació de paquets</b>	<b>29</b>
<b>8. Personalització de la instal·lació de paquets</b>	<b>29</b>
8.1 Fonts dels paquets	29
8.1.1 Distribució, àrees d'arxiu i mode	29
8.1.2 Miralls de distribució	30
8.1.3 Miralls de distribució utilitzats en temps de construcció	30
8.1.4 Miralls de distribució utilitzats en temps d'execució	30
8.1.5 Repositoris addicionals	30
8.2 Selecció dels paquets a instal·lar	31
8.2.1 Llistes de paquets	31
8.2.2 Ús dels metapaquets	31
8.2.3 Llistes locals de paquets	32
8.2.4 Llistes locals de paquets per a l'etapa binary	32
8.2.5 Generar llistes de paquets	32
8.2.6 Ús de condicionals dins de les llistes de paquets	33
8.2.7 Eliminar paquets durant la instal·lació	33
8.2.8 Tasques d'escriptori i llenguatge	33
8.2.9 Tipus i versió del nucli	34
8.2.10 Nuclis personalitzats	34
8.3 Instal·lació de paquets modificats o de tercers	35
8.3.1 Fer servir packages.chroot per a instal·lar paquets personalitzats	35
8.3.2 Fer servir un repositori APT per a instal·lar paquets personalitzats	36
8.3.3 Paquets personalitzats i APT	36

8.4 Configurar APT en temps de construcció . . . . .	36	<b>Personalització de la imatge binària</b>	<b>49</b>
8.4.1 Elegir apt o aptitude . . . . .	36	<b>11. Personalització de la imatge binària</b>	<b>49</b>
8.4.2 L'ús d'un proxy amb APT . . . . .	36	11.1 Carregadors d'arrencada . . . . .	49
8.4.3 Afinar APT per a estalviar espai . . . . .	37	11.2 metadades ISO . . . . .	49
8.4.4 Passar opcions per a apt o aptitude . . . . .	37	<b>Personalització de l'instal·lador de debian</b>	<b>50</b>
8.4.5 APT pinning . . . . .	38	<b>12. Personalització de l'instal·lador de debian</b>	<b>50</b>
<b>Personalització dels continguts</b>	<b>39</b>	12.1 Tipus d'instal·lador de Debian . . . . .	50
<b>9. Personalització dels continguts</b>	<b>39</b>	12.2 Personalització de l'instal·lador de Debian amb	
9.1 Includes . . . . .	39	preconfiguració . . . . .	51
9.1.1 Live/chroot local includes . . . . .	39	12.3 Personalitzar el contingut de l'instal·lador de Debian	51
9.1.2 Binary local includes . . . . .	40	<b>Projecte</b>	<b>52</b>
9.2 Scripts ganxo (Hooks) . . . . .	40	<b>Contribuir al projecte</b>	<b>53</b>
9.2.1 Live/chroot local hooks . . . . .	40	<b>13. Contribuir al projecte</b>	<b>53</b>
9.2.2 Scripts ganxo durant l'arrencada . . . . .	40	13.1 Fer canvis . . . . .	53
9.2.3 Binary local hooks . . . . .	40	<b>Informar dels errors</b>	<b>55</b>
9.3 Preconfiguració de les preguntes de Debconf . . . . .	41	<b>14. Informar dels errors</b>	<b>55</b>
<b>Personalització dels comportaments en temps</b>		14.1 Problemes coneguts . . . . .	55
<b>d'execució</b>	<b>42</b>	14.2 Reconstruir des de zero . . . . .	55
<b>10. Personalització dels comportaments en temps d'execució</b>	<b>42</b>	14.3 Fer servir paquets actualitzats . . . . .	55
10.1 Personalitzar l'usuari en viu . . . . .	42	14.4 Recopilar informació . . . . .	55
10.2 Personalització de l'entorn local i el llenguatge . . . . .	42	14.5 Aïllar el cas que falla, si és possible . . . . .	56
10.3 Persistència . . . . .	44	14.6 Utilitzar el paquet correcte per a informar de l'error	56
10.3.1 El fitxer persistence.conf . . . . .	45	14.6.1 A l'hora de construir mentre bootstrapping . . . . .	56
10.3.2 Utilitzar diversos medis persistents . . . . .	45		
10.4 Persistència amb xifratge . . . . .	46		

14.6.2 A l'hora de construir, durant la instal·lació de paquets . . . . .	57	<b>Exemples</b>	<b>65</b>
14.6.3 En el moment d'arrencar . . . . .	57		
14.6.4 En temps d'execució . . . . .	57	<b>Exemples</b>	<b>66</b>
14.7 Fer la recerca . . . . .	57		
14.8 On informar dels errors . . . . .	57	<b>18. Exemples</b>	<b>66</b>
		18.1 Ús dels exemples . . . . .	66
		18.2 Tutorial 1: Una imatge per defecte . . . . .	66
		18.3 Tutorial 2: Una utilitat de navegador web . . . . .	67
		18.4 Tutorial 3: Una imatge personalitzada . . . . .	67
		18.4.1 Primera revisió . . . . .	67
		18.4.2 Segona revisió . . . . .	68
		18.5 Un client per a un quiosc VNC . . . . .	69
		18.6 Una imatge bàsica per a un dispositiu USB de 128MB . . . . .	70
		18.7 Un escriptori GNOME localitzat i amb instal·lador . . . . .	71
<b>Estil de Codi</b>	<b>59</b>	<b>Apèndix</b>	<b>73</b>
<b>15. Estil de Codi</b>	<b>59</b>		
15.1 Compatibilitat . . . . .	59	<b>Style guide</b>	<b>74</b>
15.2 Indentació . . . . .	59		
15.3 Ajust de línia . . . . .	59	<b>19. Guia d'estil</b>	<b>74</b>
15.4 Variables . . . . .	59	19.1 Instruccions per als autors . . . . .	74
15.5 Miscel·lània . . . . .	60	19.1.1 Característiques lingüístiques . . . . .	74
		19.1.2 Procediments . . . . .	75
		19.2 Directrius per als traductors . . . . .	77
		19.2.1 Consells de traducció . . . . .	77
<b>Procediments</b>	<b>61</b>	<b>SiSU Metadata, document information</b>	<b>79</b>
<b>16. Procediments</b>	<b>61</b>		
16.1 Publicacions majors . . . . .	61		
16.2 Publicacions puntuals . . . . .	61		
16.2.1 Última publicació puntual d'una versió de Debian . . . . .	61		
16.2.2 Plantilla per a anunciar una publicació puntual . . . . .	61		
<b>Repositoris Git</b>	<b>63</b>		
<b>17. Repositoris Git</b>	<b>63</b>		
17.1 Gestió de múltiples repositoris . . . . .	63		

1	<b>Manual de Live Systems</b>
---	-------------------------------

2	<b>Sobre aquest manual</b>
---	----------------------------



## 3 Sobre aquest manual

### 4 1. Sobre aquest manual

5 Aquest manual serveix com a punt d'accés únic a tota la documentació relacionada amb el Live Systems Project i en particular s'aplica al programari produït pel projecte per a la versió Debian 8.0 "jessie". Una versió actualitzada es pot trobar sempre a <http://live-systems.org/>

6 Si bé *live-manual* es centra principalment en ajudar a construir un sistema viu i no en temes dels usuaris finals, un usuari final pot trobar informació útil en aquestes seccions: «**Conceptes bàsics**» abasta la descàrrega d'imatges prefabricades i la preparació de les imatges per a ser arrencades des dels dispositius o la xarxa, ja sigui utilitzant el servei de construcció d'imatges web o executant *live-build* directament en el sistema. «**Personalització dels comportaments en temps d'execució**» descriu algunes de les opcions que es poden especificar durant l'arrencada del sistema, com ara la selecció de la disposició del teclat, la configuració regional i l'ús de la persistència.

7 Algunes de les ordres esmentades en el text s'han d'executar amb privilegis de superusuari que es poden obtenir esdevenint l'usuari root amb `su` o mitjançant l'ús de `sudo`. Per a distingir entre les ordres que poden ser executades per un usuari sense privilegis i aquelles que requereixen privilegis de superusuari, s'anteposa \$ o # respectivament. Aquest símbol no és part de l'ordre.

### 8 1.1 Per als impacients

9 Si bé creiem que tot el que hi ha en aquest manual és important, si més no, per a alguns dels nostres usuaris, ens adonem que hi ha una gran quantitat de material per a cobrir i que és possible

que es vulgui experimentar l'èxit amb el programari aviat, abans d'aprofundir en els detalls. Per tant, us recomanem llegir en el següent ordre.

En primer lloc, llegir aquest capítol, «**Sobre aquest manual**», des del principi i acabant amb els «**Termes**». A continuació, saltar als tres tutorials abans dels «**Exemples**», secció pensada per a ensenyar com fer la construcció d'una imatge i alguns aspectes bàsics de la personalització. Llegir en primer lloc «**Ús dels exemples**» seguit pel «**Tutorial 1: Una imatge per defecte**», «**Tutorial 2: Una utilitat de navegador web**» i finalment, «**Tutorial 3: Una imatge personalitzada**». Al final d'aquests tutorials, es tindrà una idea del que es pot fer amb els sistemes en viu.

Us animem a tornar i a fer un estudi del manual en profunditat, la propera lectura pot ser «**Conceptes bàsics**», fregant o saltant «**Construir una imatge netboot**», i acabant amb la lectura de la «**Visió general de la personalització**» i els capítols que segueixen. En aquest punt, esperem que estigueu ben emocionats pel que es pot fer amb els sistemes en viu i motivats per llegir la resta del manual, de principi a fi.

### 1.2 Termes

- **Sistema viu** : Un sistema operatiu que pot arrencar sense necessitat d'instal·lació en un disc dur. Els sistemes vius no alteren el sistema operatiu local(s) o els fitxer(s) ja instal·lats al disc dur de l'ordinador a menys que així se'ls ho indiqui. Els sistemes vius normalment s'inicien des de dispositius, com ara CDs, DVDs o memòries USB. Alguns també poden arrencar des de la xarxa (amb les imatges netboot, veure «**Construir una imatge netboot**»), o mitjançant internet (amb el parametre `fetch=URL`, veure «**Webbooting**»).
- **Medi en viu** : A diferència de sistema en viu, el medi en viu es

refereix al CD, DVD o memòria USB on es copia el fitxer binari 26  
 produït per *live-build* i utilitzat per a arrencar el sistema en viu.  
 Més àmpliament, el terme també es refereix a qualsevol lloc on  
 resideix el fitxer binari als efectes d'iniciar el sistema en viu, com  
 ara la ubicació dels fitxers per a arrencar en xarxa.

- 15 • **Live Systems Project** : El projecte que manté, entre altres, els  
 paquets *live-boot*, *live-build*, *live-config*, *live-tools* i *live-manual*.
- 16 • **Sistema amfitrió** : L'entorn utilitzat per a crear el sistema en viu.
- 17 • **Sistema objectiu** : L'entorn que s'utilitza per a executar el  
 sistema en viu.
- 18 • **live-boot** : Una col·lecció de scripts per a arrencar els sistemes  
 vius.
- 19 • **live-build** : Una col·lecció d'scripts utilitzats per a construir  
 sistemes en viu personalitzats.
- 20 • **live-config** : Una col·lecció de scripts utilitzats per a configurar  
 un sistema en viu durant el procés d'arrencada.
- 21 • **live-tools** : Una col·lecció d'scripts addicionals que s'utilitzen per  
 a realitzar tasques útils en un sistema viu en execució.
- 22 • **live-manual** : Aquest document és mantingut en un paquet  
 anomenat *live-manual*
- 23 • **Instal·lador de Debian (d-i)** : El sistema oficial d'instal·lació de  
 la distribució Debian.
- 24 • **Paràmeters d'arrencada** : Els paràmetres que es poden  
 introduir a l'indicador del carregador d'arrencada per a influir en  
 el nucli o *live-config*
- 25 • **chroot** : El programa *chroot*, *chroot(8)*, ens permet executar  
 diferents instàncies d'un entorn GNU/Linux a la vegada en un  
 sol sistema sense reiniciar.

- **Imatge binària** : Un fitxer que conté el sistema en viu, com ara  
*live-image-i386.hybrid.iso* o *live-image-i386.img*.

- **Distribució objectiu** : La distribució en què es basa el sistema 27  
 en viu. Que pot diferir de la distribució del sistema amfitrió.

- **stable/testing/unstable** : La distribució **stable** , actualment 28  
 anomenada **wheezy** , conté l'última distribució de Debian  
 llançada oficialment. La distribució **testing** , temporalment  
 anomenada **jessie** , és l'àrea d'assaig per a la pròxima versió  
**stable** . Un avantatge important de l'ús d'aquesta distribució  
 és que té versions més recents del programari en relació  
 amb l'edició **stable** . La distribució **unstable** , permanentment  
 anomenada **sid** , és on es produeix el desenvolupament actiu  
 de Debian. En general, aquesta distribució és utilitzada pels  
 desenvolupadors i els que els agrada viure en risc. Al llarg del  
 manual, es tendeix a utilitzar els seus noms en clau, com ara  
**jessie** o **sid** , ja que és el que fan servir les pròpies eines.

### 1.3 Autors 29

Llistat d'autors (en ordre alfabètic) 30

- Ben Armstrong 31
- Brendan Sleight 32
- Carlos Zuferri 33
- Chris Lamb 34
- Daniel Baumann 35
- Franklin Piat 36
- Jonas Stein 37
- Kai Hendry 38

- Marco Amadori
- Mathieu Geli
- Matthias Kirschner
- Richard Nelson
- Trent W. Buck

## 1.4 Contribuir a aquest document

Aquest manual està pensat com un projecte comunitari i totes les propostes per a millorar-lo i les contribucions són molt benvingudes. Veure la secció **Contribuir al projecte** per a obtenir informació detallada sobre com obtenir la clau i fer bons commits.

### 1.4.1 Aplicar canvis

Per tal de realitzar canvis en el manual anglès s'ha d'editar els fitxers adequats a `manual/en/` però abans de presentar una contribució, s'ha de previsualitzar el treball. Per a previsualitzar el *live-manual*, assegurar-se que s'han instal·lat els paquets necessaris per a la seva construcció mitjançant l'execució de:

```
# apt-get install make po4a ruby ruby-nokogiri sisu-complete
```

Es pot crear el *live-manual* des del directori de nivell superior del arbre Git mitjançant l'execució de:

39

```
$ make build
```

Tenint en compte que es necessita un cert temps per construir el manual en tots els idiomes suportats, els autors poden utilitzar un dels mètodes abreujats per fer revisions ràpides de la nova documentació que han afegit al manual en anglès. Amb `PROOF=1` es construeix *live-manual* en format html, però sense els fitxers html segmentats, i amb `PROOF=2` es construeix *live-manual* en format pdf, però només el retrat A4 i carta. És per això que amb l'ús de qualsevol de les possibilitats `PROOF=` es pot estalviar una quantitat considerable de temps, per exemple:

51

52

```
$ make build PROOF=1
```

Quan es revisa una de les traduccions, és possible construir un sol idioma mitjançant l'execució de, per exemple:

53

54

```
$ make build LANGUAGES=de
```

També es possible crear per tipus de document, per exemple:

55

56

```
$ make build FORMATS=pdf
```

O combinar tot dos, per exemple:

57

58

```
$ make build LANGUAGES=de FORMATS=html
```

Després de revisar el treball i assegurar-se que tot està bé, no fer un `make commit` a menys que s'actualitzin les traduccions al mateix temps, i en aquest cas, no barrejar els canvis al manual anglès i les traduccions en el mateix commit, fer-ne un altre separat per a cada canvi. Veure la secció [«Traducció»](#) per a més detalls.

## 1.4.2 Traducció

Per a traduir *live-manual*, seguir aquests passos, depenent de si s'està començant una traducció des de zero o si es segueix treballant en una ja existent:

- Començar una nova traducció des de zero
  - Traduir els fitxers **about\_manual.ssi.pot**, **about\_project.ssi.pot** i **index.html.in.pot** de `manual/pot/` a la vostra llengua amb el vostre editor favorit (per exemple *poedit*). Enviar els fitxers `.po` traduïts a la llista de correu per comprovar la seva integritat. La comprovació d'integritat de *live-manual* garanteix que els fitxers `.po` són traduïts al 100% però també detecta possibles errors.
  - Un cop comprovats, per a activar una nova llengua al autobuild, només cal afegir els fitxers inicials traduïts a `manual/po/${LANGUAGE}/` i executar `make commit`. I llavors editar `manual/_sisu/home/index.html` afegint el nom de la llengua i el seu nom en anglès entre parèntesis.
- Continuar amb una traducció ja començada
  - Una vegada que s'ha afegit la nova llengua, es pot continuar traduint la resta de fitxers `.po` dins de `manual/-po/${LANGUAGE}/` a l'atzar amb l'editor favorit (com per exemple *poedit*).
  - No oblidar que es necessari fer un `make commit` per a garantir que els manuals traduïts s'actualitzin a partir

dels fitxers `.po` i llavors es poden revisar els canvis executant `make build` abans de `git add ., git commit -m "Translating..."` i `git push`. Recordar que `make build` pot trigar una quantitat considerable de temps, però es poden revisar els idiomes de forma individual com s'explica a la secció [«Aplicar canvis»](#).

Després d'executar `make commit` es podrà veure bastant text a la pantalla. Bàsicament són missatges informatius sobre l'estat del procés i també algunes pistes sobre el que es pot fer per a millorar *live-manual*. Si no es veu cap error fatal, generalment es pot procedir i enviar la contribució.

*live-manual* ve amb dues utilitats que poden ser de gran ajuda pels traductors a l'hora de trobar missatges sense traduir i difusos. La primera és "make translate". Aquesta activa un script que diu en detall quants missatges sense traduir hi ha a cada fitxer `.po`. La segona, "make fixfuzzy", només actua sobre els missatges difusos però ajuda a trobar-los i corregir-los un per un.

Tenir en compte que tot i que aquestes utilitats poden ser molt útils per a fer traduccions en la línia d'ordres, l'ús d'una eina especialitzada com *poedit* és la manera recomanada de fer la tasca. També és una bona idea llegir la documentació de localització de debian (l10n) i, específicament dins *live-manual*, les [«Directrius per a traductors»](#).

**Nota:** Es pot utilitzar `make clean` per a netejar l'arbre git abans de fer un push. Aquest pas no és obligatori, gràcies al fitxer `.gitignore`, però és una bona pràctica per a evitar enviar fitxers de forma involuntària.

72	<b>Sobre el Live Systems Project</b>	85	<b>2.1.2 Per què crear el nostre pròpi sistema viu?</b>	
73	<b>2. Sobre el Live Systems Project</b>	86	Debian és el sistema operatiu universal: Debian té un sistema viu per a mostrar arreu i per a representar acuradament el sistema Debian amb els següents avantatges:	
74	<b>2.1 Motivació</b>		<ul style="list-style-type: none"> <li>• Es tracta d'un subprojecte de Debian. 87</li> <li>• Reflecteix l'estat (actual) d'una distribució. 88</li> <li>• Funciona en tantes arquitectures com és possible. 89</li> <li>• Es tracta només de paquets Debian sense modificacions. 90</li> <li>• No conté paquets que no són a l'arxiu de Debian. 91</li> <li>• S'utilitza un nucli Debian sense alteracions i sense pedaços addicionals. 92</li> </ul>	
75	<b>2.1.1 Què passa amb els sistemes vius actuals</b>		<b>2.2 Filosofia</b>	93
76	Quan el Live Systems Project va començar, ja hi havia diversos sistemes vius basats en Debian disponibles i que estaven fent una gran feina. Des de la perspectiva de Debian la majoria d'ells tenien un o més dels desavantatges següents:		<b>2.2.1 Només paquets Debian sense modificacions de la secció "main"</b>	94
77	<ul style="list-style-type: none"> <li>• No són projectes de Debian i per tant no tenen suport des de Debian.</li> </ul>		Només farem servir els paquets del repositori de Debian de la secció "main". La secció non-free no és part de Debian i per tant no es pot utilitzar per a les imatges oficials del sistema viu. 95	
78	<ul style="list-style-type: none"> <li>• Es barregen diferents distribucions, per exemple <b>testing</b> i <b>unstable</b>.</li> </ul>		No canviarem cap paquet. Cada vegada que hàgim de canviar alguna cosa, ho farem en coordinació amb el mantenidor del paquet a Debian. 96	
79	<ul style="list-style-type: none"> <li>• Només donen suport a i386.</li> </ul>		Com a excepció, els nostres propis paquets, com ara <i>live-boot</i> , <i>live-build</i> o <i>live-config</i> poden ser utilitzats temporalment des del nostre propi repositori per raons de desenvolupament (per exemple, per a crear instantànies de desenvolupament). Aquests paquets es pujaran a Debian de forma regular. 97	
80	<ul style="list-style-type: none"> <li>• Es modifiqui el comportament i/o l'aparença dels paquets, per a estalviar espai.</li> </ul>			
81	<ul style="list-style-type: none"> <li>• S'inclouen paquets de fora de l'arxiu de Debian.</li> </ul>			
82	<ul style="list-style-type: none"> <li>• Inclouen nuclis personalitzats amb pedaços addicionals que no són part de Debian.</li> </ul>			
83	<ul style="list-style-type: none"> <li>• Són grans i lents a causa de la seva mida i per tant no aptes com a sistemes de rescat.</li> </ul>			
84	<ul style="list-style-type: none"> <li>• No estan disponibles en diferents formats, per exemple, CD, DVD, memòries USB i imatges netboot.</li> </ul>			

### 2.2.2 Paquets del sistema viu sense cap configuració

En aquesta fase no es publicarà o s'instal·larà cap configuració alternativa o d'exemple. Tots els paquets són utilitzats en la seva configuració per defecte, tal com són després d'una instal·lació normal de Debian.

Cada vegada que ens calgui una configuració per defecte diferent, ho farem en coordinació amb el mantenidor del paquet Debian.

S'hi inclou un sistema per a configurar paquets mitjançant debconf que permet instal·lar paquets configurats de forma personalitzada dins d'una imatge en viu, però per a les **imatges en viu prefabricades** només utilitzarem una configuració per defecte, a menys que sigui absolutament necessari per a poder treballar en l'entorn en viu. Sempre que sigui possible, preferim adaptar els paquets a l'arxiu de Debian perquè funcionin en un sistema en viu abans que realitzar canvis en la cadena d'eines en viu o en **les configuracions de les imatges prefabricades**. per a obtenir més informació, veure **Visió general de la personalització**.

## 2.3 Contacte

- **Llista de correu** : El contacte principal del projecte és la llista de correu a <https://lists.debian.org/debian-live/>. Es pot enviar un correu directament a [debian-live@lists.debian.org](mailto:debian-live@lists.debian.org) Els arxius de la llista són a <https://lists.debian.org/debian-live/>.
- **IRC** : Un nombre d'usuaris i desenvolupadors estan presents al canal #debian-live a irc.debian.org (OFTC). Quan es pregunta al IRC, s'ha de tenir paciència esperant una resposta, si no hi ha cap resposta, es pot enviar un correu a la llista.
- **BTS** : El **Informar dels errors**.

## Usuari

## Instal·lació

### 3. Instal·lació

#### 3.1 Requisits

La construcció d'imatges en viu té molts pocs requisits.

- Accés de superusuari (root)
- Una versió actualitzada de *live-build*
- Una shell compatible amb POSIX, com ara *bash* o *dash*
- *debootstrap* o *cdebootstrap*
- Linux 2.6.x o superior.

Tenir en compte que no cal usar Debian o una distribució derivada de Debian ja que *live-build* funcionarà en gairebé qualsevol distribució amb els requisits anteriors.

#### 3.2 Instal·lació de live-build

Es pot instal·lar *live-build* en un nombre de maneres diferents:

- Des del repositori Debian
- A partir del codi font
- A partir d'instantànies

Si s'utilitza Debian, la manera recomanada és instal·lar *live-build* des del repositori de Debian.

##### 3.2.1 Des del repositori de Debian

Només cal instal·lar *live-build* com qualsevol altre paquet:

```
# apt-get install live-build
```

##### 3.2.2 A partir del codi font

*live-build* es desenvolupa utilitzant el sistema de control de versions Git. En els sistemes basats en Debian, això és proporcionat pel paquet *git*. Per a comprovar l'últim codi, executar:

```
$ git clone git://live-systems.org/git/live-build.git
```

Es pot construir i instal·lar un paquet Debian pròpi mitjançant l'execució de:

```
$ cd live-build
$ dpkg-buildpackage -b -uc -us
$ cd ..
```

Ara instal·lar qualsevol dels fitxers *.deb* recent construïts que ens interessin, per exemple,

```
# dpkg -i live-build_3.0-1_all.deb
```

Es pot instal·lar *live-build* directament al sistema mitjançant l'execució de:

```
# make install
```



i desinstal·lar amb:

```
# make uninstall
```

### 3.2.3 A partir d'instantànies

Si no es desitja construir o instal·lar *live-build* a partir de les fonts, es pot utilitzar les instantànies. Aquestes es construeixen automàticament a partir de l'última versió del Git, i estan disponibles a <http://live-systems.org/debian/>.

## 3.3 Instal·lació de live-boot i live-config

**Nota:** No cal instal·lar *live-boot* o *live-config* en el sistema per a crear sistemes personalitzats en viu. No obstant, això no farà cap mal i és útil per a fins de referència. Si només es vol la documentació, ara es poden instal·lar els paquets *live-boot-doc* i *live-config-doc* per separat.

### 3.3.1 Des del repositori de Debian

Tots dos, *live-boot* i *live-config*, estan disponibles al arxiu de Debian, d'una manera similar a [«Instal·lació de live-build»](#).

### 3.3.2 A partir del codi font

Per a utilitzar les darreres fonts del git, es pot seguir el procés següent. Assegurar-se d'estar familiaritzat amb els termes esmentats a [«Termes»](#).

- Clonar les fonts de *live-boot* i *live-config*

```
$ git clone git://live-systems.org/git/live-boot.git
$ git clone git://live-systems.org/git/live-config.git
```

Consultar les pàgines del manual de *live-boot* i *live-config* per a més detalls sobre la seva personalització si aquesta és la raó per a construir aquests paquets a partir de les fonts.

- Crear els fitxers .deb de *live-boot* i *live-config*

S'ha de crear ja sigui en la distribució de destinació o en un chroot que contingui la plataforma de destinació: és a dir, si el objectiu és **jessie** llavors s'ha de construir contra **jessie**.

Es pot utilitzar un constructor personal, com ara *pbuilder* o *sbuid* si es necessita crear *live-boot* per a una distribució de destinació diferent del sistema de construcció. Per exemple, per a les imatges en viu de **jessie**, crear *live-boot* en un chroot **jessie**. Si la distribució de destinació per casualitat coincideix amb la distribució del sistema de construcció, es pot construir directament en el sistema de construcció amb *dpkg-buildpackage* (proporcionat pel paquet *dpkg-dev*):

```
$ cd live-boot
$ dpkg-buildpackage -b -uc -us
$ cd ../live-config
$ dpkg-buildpackage -b -uc -us
```

- Utilitzar tots el fitxers .deb generats que calguin

Com *live-boot* i *live-config* són instal·lats per el sistema de construcció *live-build*, instal·lar els paquets en el sistema amfitrió no és suficient: s'ha de tractar els .deb generats com si fossin uns paquets personalitzats. Ja que el propòsit per a construir

aquests paquets a partir del codi font és per a provar coses noves a curt termini abans del llançament oficial, seguir els passos de **«Instal·lació de paquets modificats o de tercers»** per a incloure temporalment els paquets rellevants en la configuració. En particular, cal observar que ambdós paquets es divideixen en una part genèrica, una part de documentació i un o més back-ends. Incloure la part genèrica, només un dels back-ends que coincideixi amb la configuració, i, opcionalment, la documentació. Suposant que s'està construint una imatge en viu en el directori actual i que s'han generat tots els `.deb` per a una única versió dels dos paquets al directori superior, aquestes ordres de bash són per a copiar tots els paquets importants, incloent-hi els back-ends per defecte:

154

```
$ cp ../live-boot{_, -initramfs-tools, -doc}*.deb config/packages.chroot/  
$ cp ../live-config{_, -sysvinit, -doc}*.deb config/packages.chroot/
```

155

### 3.3.3 A partir d'instantànies

156

Es pot deixar que *live-build* utilitzi les darreres instantànies de *live-boot* i *live-config* configurant el repositori de tercers [live-systems.org](http://live-systems.org) en el directori de configuració de *live-build*.

## Conceptes bàsics

### 4. Conceptes bàsics

Aquest capítol conté una breu descripció del procés de construcció i les instruccions per a l'utilització dels tres tipus d'imatge més comunes. El tipus d'imatge més versàtil `iso-hybrid` es pot utilitzar en una màquina virtual, en un medi òptic o qualsevol altre dispositiu d'emmagatzematge USB. En certs casos especials, com s'explica més endavant, el tipus `hdd` pot ser el més adequat. El capítol conté instruccions detallades per a la construcció d'una imatge tipus `netboot`, que és una mica més complicat a causa de la configuració necessària en el servidor. Aquest és un tema una mica avançat per a algú que no està familiaritzat ja amb l'arrencada en xarxa, però s'inclou aquí perquè un cop que la configuració es porta a terme, es tracta d'una manera molt convenient per a provar i desplegar imatges per a l'arrencada en xarxa local sense la molèstia de tractar amb els dispositius de les imatges.

La secció acaba amb una breu introducció al `<webbooting>` que és, potser, la forma més fàcil d'utilitzar imatges diferents per a diferents propòsits, canviant d'una a l'altra segons sigui necessari, utilitzant internet com un mitjà.

Al llarg del capítol, sovint es fa referència als noms dels fitxers produïts per defecte per `live-build`. Si es `<descarrega una imatge prefabricada>`, els noms dels fitxers poden ser diferents.

#### 4.1 Què és un sistema viu?

Un sistema viu és un sistema operatiu que arrenca en un equip des d'un dispositiu extraïble, com un CD-ROM o una memòria USB o des d'una xarxa, a punt per a fer servir sense cap tipus d'instal·lació

en la unitat(s) habitual(s), amb una configuració automàtica feta en temps d'execució (veure `<Termes>`).

Amb live systems, és un sistema operatiu, construït per a una de les arquitectures suportades (actualment amd64 i i386). Conté les següents parts:

- **Imatge del nucli Linux**, generalment s'anomena `mlinuz*`
- **Imatge del disc RAM inicial (initrd)**: un disc RAM configurat per a l'arrencada de Linux, que conté els mòduls que possiblement es necessitaran per a muntar la imatge del sistema i algunes seqüències d'ordres per a fer-ho.
- **Imatge del sistema**: Imatge del sistema de fitxers del sistema operatiu. Normalment, s'utilitza un sistema de fitxers comprimit SquashFS per a minimitzar la mida de la imatge. Tenir en compte que és de només lectura. Així, durant l'arrencada, el sistema Debian Live utilitzarà el disc RAM i un mecanisme de "unió" per a permetre l'escriptura de fitxers en el sistema en funcionament. No obstant això, totes les modificacions es perdran al apagar l'equip si no és que s'utilitza la persistència opcional (vegeu `<Persistència>`).
- **Carregador d'arrencada**: Una petita peça de codi dissenyat per a arrencar des del medi triat, possiblement presentant un indicador d'arrencada o un menú per a permetre la selecció d'opcions/configuració. Carrega el nucli de Linux i el seu `initrd` per a funcionar amb un sistema de fitxers del sistema associat. Es poden utilitzar diverses solucions, en funció del medi de destinació i el format del sistema de fitxers que conté els components esmentats anteriorment: `isolinux` per a arrencar des de CD o DVD en format ISO9660, `syslinux` per a una unitat USB o HDD que s'iniciarà des de particions VFAT, `extlinux` per a particions `ext2/3/4` i `btrfs`, `pxelinux` per a PXE netboot, GRUB per a particions `ext2/3/4`, etc.

169 Es pot utilitzar *live-build* per a construir la imatge del sistema amb especificacions pròpies, configurar un nucli de Linux, el *initrd*, i un carregador d'arrencada per a executar-los, tot això en un format depenent del medi (imatge ISO9660, imatge de disc, etc.).

## 170 4.2 Descàrrega d'imatges prefabricades

171 Si bé l'objectiu d'aquest manual és el desenvolupament i la construcció d'imatges en viu pròpies, es pot simplement voler provar una de les nostres imatges prefabricades, ja sigui com una introducció al seu ús o a la construcció d'imatges personalitzades. Aquestes imatges estan construïdes utilitzant el nostre **<repositori git live-images>** i les versions oficials estables es publiquen a **<<https://www.debian.org/CD/live/>>**. A més, les versions antigues i les futures, i les imatges no oficials que contenen microprogramari i controladors no lliures estan disponibles a **<<http://live-systems.org/cdimage/release/>>**.

## 172 4.3 Ús del servei de construcció d'imatges en viu web

173 Com un servei a la comunitat, oferim un servei web capaç de construir imatges en viu a **<<http://live-build.debian.net/>>**. Aquest lloc es mantingut sobre una base del millor esforç. És a dir, encara que ens esforcem per a mantenir-lo al dia i que estigui operatiu en tot moment, i de fer anuncis d'interrupcions importants del servei, no podem garantir una disponibilitat del 100% o una creació d'imatges ràpida. Tanmateix, el servei pot tenir, de tant en tant, problemes que triguin algun temps en resoldre's. Si es té problemes o preguntes sobre aquest servei, posar-se en **<contacte amb nosaltres>**, proporcionant l'enllaç a la pàgina web de la imatge construïda.

### 4.3.1 Ús i advertències sobre el servei web

La interfície web actualment no pot prevenir l'ús de combinacions d'opcions no vàlides, i en particular, quan el canvi d'una opció que normalment (és a dir, utilitzant *live-build* directament) canviaria els valors predeterminats d'altres opcions que figuren en el formulari de la web, el constructor web no canvia aquests valors predeterminats. En particular, si es canvia `--architectures` del valor per defecte `i386` a `amd64`, s'ha de canviar l'opció corresponent `--linux-flavours` del valor per defecte `486` a `amd64`. Veure la pàgina del manual `lb_config` per a la versió de *live-build* instal·lada al constructor web per a més detalls. El nombre de la versió de *live-build* apareix a la part inferior de la pàgina web.

El càlcul de temps donat pel constructor web és només una estimació aproximada i pot no reflectir acuradament el temps que realment es necessita per a finalitzar la construcció d'una imatge. Tenir en compte que aquesta estimació tampoc s'actualitza tal i com va passant el temps. Cal tenir una mica de paciència. No tornar a carregar la pàgina després de enviar la proposta de construcció, ja que això pot tornar a reenviar una nova petició amb els mateixos paràmetres. Posar-se **<en contacte amb nosaltres>** si no es rep la notificació de la construcció un cop que s'estigui segur que s'ha esperat prou i verificat que la notificació per correu electrònic no ha anat a parar al spam.

El servei web està limitat en el tipus d'imatges que es poden construir. Això el manté simple i eficient d'utilitzar i mantenir. Si es desitja realitzar personalitzacions que no es contemplen en la interfície web, a la resta d'aquest manual s'explica com crear imatges pròpies amb *live-build*.

## 4.4 Primers passos: construcció d'una imatge ISO híbrida

Independentment del tipus d'imatge, s'haurà de fer els mateixos passos bàsics per a construir una imatge cada vegada. Com a primer exemple, crear un directori de treball, canviar a aquest directori i executar la següent seqüència d'ordres *live-build* per a crear una imatge ISO híbrida de base que conté només el sistema per defecte de Debian sense X.org. És adequat per a gravar en un CD o DVD, i també per a copiar en una memòria USB.

El nom del directori de treball és absolutament indiferent, però si es fa un cop d'ull als exemples utilitzats a *live-manual*, és una bona idea utilitzar un nom que ajudi a identificar la imatge amb que s'està treballant en cada directori, especialment si s'està treballant o experimentant amb diferents tipus d'imatges. En aquest cas, anem a construir un sistema per defecte així que l'anomenarem, per exemple, *live-default*.

```
$ mkdir live-default && cd live-default
```

Aleshores, executar l'ordre `lb config`. Això crearà una jerarquia «config/» en el directori actual per a ser utilitzada per altres ordres:

```
$ lb config
```

Aquí no es passa cap paràmetre a aquestes ordres, per tant s'utilitzaran les opcions per defecte. Veure «**L'ordre lb config**» per a més detalls.

Ara que la jerarquia «config/» ja existeix, crear la imatge amb

l'ordre `lb build`:

```
# lb build
```

Aquest procés tardarà una mica, depenent de la velocitat del ordinador i de la connexió de la xarxa. Quan hagi acabat, ha d'haver un fitxer imatge `live-image-i386.hybrid.iso`, a punt per a ser utilitzat, en el directori actual.

**Nota:** Si s'està construint en un sistema amd64 el nom de la imatge resultant serà `live-image-amd64.hybrid.iso`. Tenir en compte aquesta convenció al llarg del manual.

## 4.5 Usar una imatge ISO híbrida en viu

Després de la construcció o la descàrrega d'una imatge ISO híbrida, que pot ser obtinguda a <https://www.debian.org/CD/live/>, el següent pas habitual és preparar el dispositiu per a l'arrencada, ja sigui medis òptics com un CD-R(W) o DVD-R(W) o una memòria USB.

### 4.5.1 Gravar una imatge ISO en un medi físic

Gravar una imatge ISO és fàcil. Simplement cal instal·lar *xorriso* i utilitzar-lo des de la línia d'ordres per a gravar la imatge. Per exemple:

```
# apt-get install xorriso
$ xorriso -as cdrecord -v dev=/dev/sr0 blank=as_needed live-image-i386.hybrid.iso
```

### 4.5.2 Copiar una imatge ISO híbrida en un dispositiu USB

Les imatges ISO preparades amb xorriso, és poden copiar directament a una memòria USB utilitzant el programa `cp` o un altre d'equivalent. Connectar una memòria USB amb una mida prou gran per al fitxer de la imatge i determinar quin dispositiu és, que d'ara endavant anomenarem `${USBSTICK}`. Aquest és el dispositiu de la memòria, com per exemple `/dev/sdb`, i no una partició, com ara `/dev/sdb1`! Es pot trobar el nom del dispositiu correcte mirant la sortida de `dmesg` després de connectar la memòria usb, o encara millor, `ls -l /dev/disk/by-id`.

Quan s'estigui segur de tenir el nom del dispositiu correcte, utilitzar l'ordre `cp` per a copiar la imatge a la memòria. **Fent això es perdran definitivament tots els continguts anteriors de la memòria usb!**

```
$ cp live-image-i386.hybrid.iso ${USBSTICK}
$ sync
```

**Nota:** L'ordre `sync` s'utilitza per a assegurar-se que totes les dades, que el nucli emmagatzema en la memòria durant la còpia de la imatge, s'escriuen en el dispositiu USB.

### 4.5.3 Utilitzar l'espai lliure en una memòria USB

Després de copiar la `live-image-i386.hybrid.iso` en un dispositiu USB, la primera partició serà utilitzada pel sistema en viu. Per a poder utilitzar l'espai que queda lliure es pot utilitzar una eina de particionament com *gparted* o *parted* per a crear una nova partició.

```
# gparted ${USBSTICK}
```

Després de crear la partició, on `${PARTITION}` és el nom de la partició, com ara `/dev/sdb2`, s'ha de crear un sistema de fitxers. Una opció possible seria `ext4`.

```
# mkfs.ext4 ${PARTITION}
```

**Nota:** Si es vol utilitzar l'espai addicional amb Windows, pel que sembla, aquest sistema operatiu normalment no pot accedir a altres particions més que a la primera. Algunes solucions a aquest problema han estat discutides a la nostra [llista de correu](#), però sembla que no hi ha respostes fàcils.

**Recordar:** Cada vegada que s'instal·li una nova `live-image-i386.hybrid.iso` al dispositiu, es perdran totes les dades perquè la taula de particions se sobreescriu amb el contingut de la imatge, de manera que es assenyat fer una còpia de seguretat de la partició addicional per a restaurar les dades de nou després d'actualitzar la imatge en viu.

### 4.5.4 Arrencar el medi en viu

La primera vegada que s'arrenqui el medi en viu, ja sigui des de CD, DVD, memòria USB, o PXE, pot ser necessària alguna petita configuració al BIOS del ordinador en primer lloc. Atès que les BIOS varien molt en les seves funcions i dreceres de teclat, no podem entrar en el tema en profunditat aquí. Algunes BIOS proporcionen una tecla per a obrir un menú de dispositius d'arrencada, que és la manera més fàcil si es troba disponible al sistema. En cas contrari, cal entrar al menú de configuració del

BIOS i canviar l'ordre d'arrencada per a situar el dispositiu del sistema en viu abans que el dispositiu d'arrencada normal.

Després d'arrencar des del dispositiu, es veurà un menú d'inici. Si es prem «entrar» el sistema s'iniciarà amb l'entrada `Live` i les seves opcions per defecte. Per a obtenir més informació sobre les opcions d'arrencada, llegir «l'ajuda» (help) al menú i també les pàgines del manual de *live-boot* i *live-config* que es troben dins del sistema en viu.

Suposant que s'ha seleccionat `Live` i s'ha arrencat una imatge d'escriptori per defecte, després que els missatges d'arrencada hagin passat s'haurà iniciat una sessió com a usuari `user` i es veurà un escriptori, a punt per a ser utilitzat. Si s'ha arrencat una imatge de només consola, com les *<imatges prefabricades>*, `standard` o `rescue` s'iniciarà una sessió com a usuari `user` i es veurà el indicador de la shell, a punt per a ser utilitzat.

## 4.6 Utilitzar una màquina virtual per a fer proves

Pot ser un gran estalvi de temps per al desenvolupament d'imatges en viu executar-les en una màquina virtual (VM). Això no està exempt d'avertiments:

- L'execució d'una màquina virtual requereix de suficient memòria RAM, tant per al sistema operatiu convidat i l'amfitrió i es recomana una CPU amb suport de maquinari per a la virtualització.
- Hi ha algunes limitacions inherents a l'execució en una màquina virtual, per exemple, rendiment de vídeo pobre, opcions limitades de maquinari emulat.
- En el desenvolupament per a un maquinari específic, no hi ha cap substitut millor que el propi maquinari.

- De tant en tant hi ha errors que només sorgeixen durant l'execució en una màquina virtual. En cas de dubte, comprovar la imatge directament al maquinari.

Sempre que es pugui treballar dins d'aquestes limitacions, examinar el programari de màquina virtual disponible i triar un que sigui adequat per a les necessitats pròpies.

### 4.6.1 Provar una imatge ISO amb QEMU

La màquina virtual més versàtil dins Debian és QEMU. Si el processador té suport de maquinari per a la virtualització, utilitzar el paquet *qemu-kvm*; la descripció del paquet *qemu-kvm* enumera breument els requisits.

Primer, instal·lar *qemu-kvm* si el processador ho suporta. Si no, instal·lar *qemu*, en aquest cas el nom del programa és *qemu* en lloc de *kvm* en els exemples següents. El paquet *qemu-utils* també és valuós per a la creació d'imatges de disc virtuals amb *qemu-img*.

```
# apt-get install qemu-kvm qemu-utils
```

Arrencar una imatge ISO és senzill:

```
$ kvm -cdrom live-image-i386.hybrid.iso
```

Veure les pàgines del manual per a més detalls

### 4.6.2 Provar una imatge ISO amb VirtualBox

Per a provar la ISO amb *virtualbox*:



```
# apt-get install virtualbox virtualbox-qt virtualbox-dkms
$ virtualbox
```

Crear una nova màquina virtual, canviar els paràmetres d'emmagatzematge per a utilitzar `live-image-i386.hybrid.iso` com unitat de CD/DVD i arrencar la màquina.

**Nota:** Per a provar sistemes vius que contenen X.org amb *virtualbox*, segurament es assenyat incloure el paquet del driver VirtualBox X.org, *virtualbox-guest-dkms* i *virtualbox-guest-x11*, en la configuració de *live-build*. En cas contrari, la resolució es limita a 800x600.

```
$ echo "virtualbox-guest-dkms virtualbox-guest-x11" >> config/package-lists<->
/my.list.chroot
```

Per tal de fer que el paquet *dkms* funcioni, s'han d'instal·lar també les capçaleres del nucli per a la variant del nucli de la imatge. En lloc d'enumerar manualment el paquet *linux-headers* correcte en la llista de paquets creat anteriorment, la selecció del paquet adequat es pot fer automàticament amb *live-build*.

```
$ lb config --linux-packages "linux-image linux-headers"
```

## 4.7 Construir i utilitzar una imatge HDD

Construir una imatge HDD és similar a una ISO híbrida en tots els aspectes, excepte que s'especifica `-b hdd`, que el nom del

fitxer resultant és `live-image-i386.img` i que no es pot gravar en medis òptics. És adequada per a arrencar des de dispositius USB, discs durs USB, i altres dispositius d'emmagatzematge portàtils. Normalment, una imatge ISO híbrida es pot utilitzar per a aquest propòsit en el seu lloc, però si el BIOS no maneja adequadament les imatges híbrides, cal utilitzar una imatge HDD.

**Nota:** si s'ha creat una imatge ISO híbrida amb l'exemple anterior, s'haurà de netejar el directori de treball amb l'ordre `lb clean` (veure <L'ordre `lb clean`>):

```
# lb clean --binary
```

Executar l'ordre `lb config` com abans, excepte que aquesta vegada especificant el tipus d'imatge HDD:

```
$ lb config -b hdd
```

Ara construir la imatge amb l'ordre `lb build`:

```
# lb build
```

Quan la construcció acabi, hauria d'haver un fitxer `live-image-i386.img` al directori actual.

La imatge binària generada conté una partició VFAT i el carregador d'arrencada `syslinux`, llestos per a ser escrits directament a una memòria USB. Un cop més, donat que l'ús d'una imatge HDD és com utilitzar una imatge ISO híbrida en un USB, seguir les instruccions de <Usar una imatge ISO híbrida en viu>, però amb



el nom de fitxer `live-image-i386.img` en lloc de `live-image-i386.hybrid.iso`.

Igualment, per a provar una imatge HDD amb Qemu, instal·lar *qemu* com s'ha descrit anteriorment a «**Provar una imatge ISO amb QEMU**». A continuació, executar `kvm` o `qemu`, segons la versió instal·lada al sistema amfitrió, especificant `live-image-i386.img` com a primer disc dur.

```
$ kvm -hda live-image-i386.img
```

## 4.8 Construir una imatge netboot

La següent seqüència d'ordres crearà una imatge netboot bàsica que conté el sistema per defecte de Debian sense X.org. És adequada per a l'arrencada en xarxa.

**Nota:** si s'ha realitzat algun dels exemples anteriors, s'haurà de netejar el directori de treball amb l'ordre `lb clean`:

```
# lb clean
```

En aquest cas concret, un `lb clean --binary` no seria suficient per a netejar les etapes necessàries. La causa d'això és que en les configuracions d'arrencada en xarxa, es necessita una configuració `initramfs` diferent que *live-build* realitza automàticament quan es construeixen imatges netboot. Ja que la creació del `initramfs` pertany a l'etapa `chroot`, fer el canvi a netboot en un directori de construcció existent significa reconstruir l'etapa `chroot` també. Per tant, s'ha de fer un `lb clean` (que eliminarà l'etapa `chroot`, també)

Executar l'ordre següent per a configurar la imatge per a arrencar en xarxa:

```
$ lb config -b netboot --net-root-path "/srv/debian-live" --net-root-server↔  
"192.168.0.2"
```

A diferència de les imatges ISO i HDD, l'arrencada en xarxa no serveix el sistema de fitxers al client, per tant els fitxers han de ser servits a través de NFS. Amb `lb config` es poden elegir diferents sistemes de fitxers de xarxa. Les opcions `--net-root-path` i `--net-root-server` especifiquen la ubicació i el servidor, respectivament, del servidor NFS on es troba la imatge del sistema de fitxers a l'hora d'arrencar. Assegurar-se que aquests s'ajusten als valors adequats per a la xarxa i el servidor.

Ara construir la imatge amb l'ordre `lb build`:

```
# lb build
```

En l'arrencada en xarxa, el client executa una petita peça de programari que normalment es troba a la EPROM de la targeta Ethernet. Aquest programa envia una petició DHCP per a obtenir una adreça IP i la informació sobre què fer a continuació. Per regla general, el següent pas és aconseguir un carregador d'arrencada de més alt nivell a través del protocol TFTP. Podria ser GRUB, pxelinux o fins i tot arrencar directament a un sistema operatiu com Linux.

Per exemple, si es descomprimeix el arxiu `live-image-i386.netboot.tar` generat al directori `/srv/debian-live`, es trobarà la imatge del sistema de fitxers a `live/filesystem.squashfs` i el nucli, `initrd` i carregador d'arrencada `pxelinux` a `tftpboot/`.

Ara hem de configurar els tres serveis al servidor per a l'arrencada en xarxa: el servidor DHCP, servidor TFTP i el servidor NFS.

#### 4.8.1 Servidor DHCP

S'ha de configurar el servidor DHCP de la xarxa per a assegurar-se que dona una adreça IP per al client del sistema d'arrencada en xarxa, i per a anunciar la ubicació del carregador d'arrencada PXE.

Heus aquí un exemple per a servir d'inspiració, escrit per al servidor ISC DHCP `isc-dhcp-server` al fitxer de configuració `/etc/dhcp/dhcpd.conf`:

```
# /etc/dhcp/dhcpd.conf - configuration file for isc-dhcp-server

ddns-update-style none;

option domain-name "example.org";
option domain-name-servers ns1.example.org, ns2.example.org;

default-lease-time 600;
max-lease-time 7200;

log-facility local7;

subnet 192.168.0.0 netmask 255.255.255.0 {
    range 192.168.0.1 192.168.0.254;
    filename "pxelinux.0";
    next-server 192.168.0.2;
    option subnet-mask 255.255.255.0;
    option broadcast-address 192.168.0.255;
    option routers 192.168.0.1;
}
```

#### 4.8.2 Servidor TFTP

Aquest serveix el nucli i el disc ram inicial per al sistema en temps d'execució.

S'ha d'instal·lar el paquet `tftpd-hpa`. Aquest pot servir tots els fitxers continguts dins d'un directori arrel, per regla general `/srv/tftp`. Per tal que es serveixin els fitxers dins de `/srv/debian-live/tftpboot`, s'ha d'executar com a superusuari la següent ordre:

```
# dpkg-reconfigure -plow tftpd-hpa
```

i omplir el nou directori del servidor tftp quan ho hàgim de fer.

#### 4.8.3 Servidor NFS

Un cop l'ordinador ha descarregat, ha arrencat el nucli de Linux i ha carregat el `initrd`, intentarà muntar la imatge del sistema de fitxers en viu a través d'un servidor NFS.

S'ha d'instal·lar el paquet `nfs-kernel-server`

Lavors, fer que la imatge del sistema de fitxers estigui disponible a través de NFS afegint una línia com la següent a `/etc/exports`:

```
/srv/debian-live *(ro,async,no_root_squash,no_subtree_check)
```

i informar al servidor NFS sobre aquesta nova exportació amb la següent ordre:

```
# exportfs -rv
```

273 La configuració d'aquests tres serveis pot ser una mica difícil. És possible que es necessiti una mica de paciència per aconseguir que tots tres funcionin plegats. Per obtenir més informació, veure el wiki de syslinux a <http://www.syslinux.org/wiki/index.php/PXELINUX> o la secció TFTP Net Booting al Manual del Instal·lador de Debian a <http://d-i.alioth.debian.org/manual/ca.i386/ch04s05.html>. Això pot ajudar, ja que els seus processos són molt similars.

#### 274 4.8.4 Com provar l'arrencada en xarxa

275 La creació d'imatges d'arrencada en xarxa es senzilla amb *live-build*, però provar les imatges en màquines físiques pot costar molt de temps.

276 Per a fer la nostra vida més fàcil, podem utilitzar la virtualització.

#### 277 4.8.5 Qemu

278 • Instal·lar *qemu*, *bridge-utils*, *sudo*.

279 Editar `/etc/qemu-ifup`:

280

```
#!/bin/sh
sudo -p "Password for $0:" /sbin/ifconfig $1 172.20.0.1
echo "Executing /etc/qemu-ifup"
echo "Bringing up $1 for bridged mode..."
sudo /sbin/ifconfig $1 0.0.0.0 promisc up
echo "Adding $1 to br0..."
sudo /usr/sbin/brctl addif br0 $1
sleep 2
```

281 Descarregar o crear un `grub-floppy-netboot`.

Llançar *qemu* amb `"-net nic,vlan=0 -net tap,vlan=0,ifname=tun0"`<sup>282</sup>

## 4.9 Webbooting

283

Webbooting és una manera convenient d'aconseguir i arrencar sistemes vius utilitzant internet com un mitjà. Els requisits per fer webbooting són molt pocs. D'una banda, es necessita un dispositiu amb un carregador d'arrencada, un disc ram inicial i un nucli. D'altra banda, un servidor web per emmagatzemar els fitxers squashfs que contenen el sistema de fitxers.

284

### 4.9.1 Obtenir els fitxers webboot

285

Com de costum, es pot construir les imatges un mateix o utilitzar els fitxers prefabricats, que estan disponibles a la pàgina principal del projecte a <http://live-systems.org/>. Les imatges prefabricades són adients per fer proves inicials fins que un pugui afinar les seves pròpies necessitats. Si ja s'ha construït una imatge en viu, els fitxers necessaris pel webbooting es troben al directori de construcció sota `binary/live/`. Els fitxers s'anomenen `vmlinuz`, `initrd.img` i `filesystem.squashfs`.

286

També és possible extreure els fitxers d'una imatge iso ja existent. Per tal d'aconseguir això, muntar la imatge de la següent manera:

287

```
# mount -o loop image.iso /mnt
```

288

Els fitxers es troben sota el directori `live/`. En aquest cas concret, seria `/mnt/live/`. Aquest mètode té el desavantatge que cal ser root per poder muntar la imatge. No obstant això, té l'avantatge que és fàcil de fer amb un script, i per tant es pot automatitzar.

289

Però, sens dubte, la forma més fàcil d'extreure els fitxers d'una imatge iso i pujar-los al servidor web a la vegada, és utilitzant el midnight commander o *mc*. Si es té el paquet *genisoimage* instal·lat, aquest gestor de fitxers de dos panells permet examinar el contingut d'un arxiu iso en un panell i pujar els fitxers via ftp en l'altre panell. Tot i que aquest mètode requereix fer un treball manual, no requereix privilegis de root.

gnome. Si es necessita les eines de rescat que s'inclouen en el segon tipus d'imatge, potser per a una altra màquina, arrencar des de internet la de rescat.

#### 4.9.2 Arrencar imatges webboot

Mentre que alguns usuaris prefereixen utilitzar la virtualització per prova el webbooting, en aquest cas utilitzem maquinari real perquè coincideixi amb el següent cas d'ús, que només ha de ser considerat com un exemple.

Per a arrencar una imatge webboot és suficient tenir els components esmentats anteriorment, és a dir, *vmlinuz* i *initrd.img* en una memòria usb dins d'un directori anomenat *live/* i instal·lar *syslinux* com a gestor d'arrencada. Després, arrencar des de la memòria usb i escriure *fetch=URL/RUTA/-AL/FITXER* a les opcions d'arrencada. *live-boot* descarregarà l'arxiu *squashfs* i l'emmagatzemarà en la memòria ram. D'aquesta manera, és possible utilitzar el sistema de fitxers comprimit descarregat com si fos un sistema viu normal. Per exemple:

```
append boot=live components fetch=http://192.168.2.50/images/webboot/↔
filesystem.squashfs
```

**Cas d'ús:** Tenir un servidor web en el qual s'ha emmagatzemat dos arxius *squashfs*, un que conté un escriptori complet, com ara *gnome*, i un de rescat. Si es necessita un entorn gràfic per a una màquina, es pot connectar la memòria usb i arrencar la imatge

## 296 Descripció general de les eines

## 297 5. Descripció general de les eines

298 Aquest capítol conté un resum de les tres eines principals  
utilitzades en la construcció dels sistemes en viu: *live-build*,  
*live-boot* i *live-config*.

### 299 5.1 El paquet live-build

300 *live-build* és un conjunt de scripts per a crear sistemes en viu.  
Aquests scripts també s'anomenen «ordres».

301 La idea darrere de *live-build* és ser un marc que utilitza un directori  
de configuració per automatitzar completament i personalitzar tots  
els aspectes de la construcció d'una imatge en viu.

302 Molts conceptes són similars als utilitzats per a crear paquets  
Debian amb *debhelper*:

- 303 • Els scripts tenen una ubicació central per a la configuració del  
seu funcionament. Amb *debhelper* aquest és el subdirectori  
`debian/` d'un arbre de paquets. Per exemple, `dh_install`  
buscarà, entre altres, un fitxer anomenat `debian/install` per  
a determinar quins fitxers han d'existir en un paquet binari en  
particular. De la mateixa manera, *live-build* emmagatzema la  
seva configuració per complet sota un subdirectori `config/`.

- 304 • Els scripts són independents - és a dir, sempre és segur executar  
cada ordre.

305 A diferència de *debhelper*, *live-build* proporciona les eines per a  
generar un directori de configuració en esquelet. Això podria ser  
considerat similar a eines com ara *dh-make*. Per a més informació  
sobre aquestes eines, seguiu llegint, ja que la resta d'aquesta  
secció discuteix les quatre ordres més importants. Tenir en compte

que van precedides de `lb` que és una funció genèrica per a les  
ordres de *live-build*.

- **lb config** : Responsable d'inicialitzar un directori de configuració 306  
per al sistema en viu. Consultar «L'ordre lb config» per a més  
informació.
- **lb build** : Responsable d'iniciar la creació d'un sistema en viu. 307  
Consultar «L'ordre lb build» per a més informació.
- **lb clean** : Responsable d'eliminar parts de la construcció d'un 308  
sistema viu. Consultar «L'ordre lb clean» per a més informació.

#### 5.1.1 L'ordre lb config 309

Com s'ha dit a «live-build», les seqüències d'ordres que formen 310  
part de *live-build* llegeixen la seva configuració amb l'ordre `source`  
d'un únic directori anomenat `config/`. Com la construcció d'aquest  
directori a mà, seria molt costós i propens a errors, es pot utilitzar  
l'ordre `lb config` per a crear l'arbre inicial de configuració en  
esquelet.

Executar `lb config` sense arguments crea el subdirectori `config/` 311  
que s'omple amb alguns paràmetres per defecte en fitxers de  
configuració, i dos arbres de subdirectoris en esquelet que  
s'anomenen `auto/` i `local/`.

```
312 $ lb config
[2014-04-25 17:14:34] lb config
P: Updating config tree for a debian/wheezy/i386 system
```

Utilitzar `lb config` sense cap tipus d'arguments seria convenient 313  
per als usuaris que necessiten una imatge molt bàsica, o  
que tinguin la intenció de proporcionar una configuració més

completa més tard mitjançant `auto/config` (Veure <Gestió d'una configuració> per a més detalls).

Normalment, s'haurà d'especificar algunes opcions. Per exemple, per a especificar quin gestor de paquets utilitzar durant la construcció de la imatge:

```
$ lb config --apt aptitude
```

És possible especificar diverses opcions, com ara:

```
$ lb config --binary-images netboot --bootappend-live "boot=live components↵
hostname=live-host username=live-user" ...
```

Una llista completa d'opcions està disponible a la pàgina del manual `lb_config`.

### 5.1.2 L'ordre `lb build`

L'ordre `lb build` llegeix la configuració del directori `config/`. A continuació, executa les ordres de nivell inferior necessàries per a construir el sistema en viu.

### 5.1.3 L'ordre `lb clean`

L'ordre `lb clean` s'encarrega d'eliminar diverses parts d'una construcció per a que altres construccions posteriors puguin començar des d'un estat net. Per defecte, es netegen les etapes `chroot`, `binary` i `source`, però la caché es manté intacta. A més, es poden netejar etapes individuals. Per exemple, si s'han fet canvis que només afecten a la fase `binary`, utilitzar `lb clean --binary`

abans de construir un nou binary. Si els canvis modifiquen el bootstrap i/o la caché de paquets, per exemple, canvis en les opcions `--mode`, `--architecture` o `--bootstrap`, s'ha d'utilitzar `lb clean --purge`. Veure la pàgina del manual de `lb_clean` per a una llista completa d'opcions.

## 5.2 El paquet `live-boot`

*live-boot* és un conjunt de scripts per a proporcionar hooks a *initramfs-tools*, que s'utilitzen per a generar un `initramfs` capaç d'arrencar sistemes vius, com ara els creats per *live-build*. Això inclou les ISOs dels sistemes en viu, netboot tarballs i imatges per a memòries USB.

En el moment d'arrencar, buscarà medis de només lectura que continguin un directori `/live/` on s'emmagatzema un sistema de fitxers arrel (sovint una imatge de un sistema de fitxers comprimit `squashfs`). Si el troba, crearà un entorn d'escriptura, utilitzant `aufs`, per a que puguin arrencar sistemes com Debian o similars.

Més informació sobre `ramfs` inicial a Debian es pot trobar al Debian Linux Kernel Handbook <<http://kernel-handbook.alioth.debian.org/>> al capítol sobre `initramfs`.

## 5.3 El paquet `live-config`

*live-config* consta dels scripts que s'executen durant l'arrencada després de *live-boot* per a configurar el sistema en viu de forma automàtica. S'ocupa de tasques com ara l'establiment de les locales, el nom d'amfitrió, la zona horària, crear l'usuari en viu, l'inhibició de tasques de cron i l'inici automàtic de sessió per a l'usuari en viu.

## 329 Gestió d'una configuració

## 330 6. Gestió d'una configuració

331 En aquest capítol s'explica com gestionar una configuració d'un sistema en viu des de la seva creació inicial, a través de revisions i versions successives de tant el programari *live-build* com de la imatge en viu en si mateixa.

### 332 6.1 Gestionar canvis en la configuració

333 Les configuracions de sistemes en viu poques vegades són perfectes al primer intent. Passar opcions a `lb config` des de la línia d'ordres pot estar bé per a construir una imatge una vegada, però és més típic revisar aquestes opcions i construir de nou fins que se'n estigui satisfet. Per a donar suport a aquests canvis, es poden utilitzar scripts `auto` que assegurin que la configuració es manté en un estat consistent.

#### 334 6.1.1 Per què utilitzar scripts `auto`? Què fan?

335 L'ordre `lb config` emmagatzema les opcions que se li passen als fitxers de `config/*`, juntament amb moltes altres opcions que estan establertes als valors per defecte. Si s'executa un cop més, `lb config` no es restablirà cap de les opcions dependents basades en les opcions inicials. Així, per exemple, si s'executa de nou `lb config` amb un nou valor per a `--binary-images`, totes les opcions que en depenen que es van omplir per al tipus de imatge per defecte ja no poden funcionar amb la nova. Aquests fitxers no estan destinats a ser llegits o editats. S'emmagatzemen els valors de més de cent opcions, i ningú pot veure les opcions que s'han especificat realment. I finalment, si s'executa `lb config` i a continuació s'actualitza *live-build* i el nom d'una opció canvia,

`config/*` encara contindrà les variables de l'opció vella que ja no són vàlides.

336 Per totes aquestes raons, els scripts `auto/*` ens fan la vida més fàcil. Són simples embolcalls per les ordres `lb config`, `lb build` i `lb clean` dissenyats per ajudar a gestionar una configuració. Només cal crear un script `auto/config` que contingui totes les opcions que es desitgin per a `lb config`, i un `auto/clean` que elimini els fitxers que continguin diversos valors de variables de configuració, i el script `auto/build` guarda un `build.log` de cada construcció. Cada vegada que s'executi l'ordre `lb` corresponent, aquests fitxers seran executats automàticament. L'ús d'aquests scripts assegurarà que la configuració sigui més senzilla de llegir i que guardi una coherència interna d'una reversió a una altra. A més a més serà més fàcil identificar i solucionar les opcions que s'han de canviar al actualitzar d'una versió de *live-build* a la següent després de llegir la documentació.

#### 337 6.1.2 Utilitzar scripts `auto` d'exemple

338 Per a més comoditat, *live-build* ve amb uns scripts d'exemple per a copiar i editar. Iniciar una nova configuració per defecte, i a continuació, copiar els exemples:

```
339 $ mkdir mylive && cd mylive && lb config
$ mkdir auto
$ cp /usr/share/doc/live-build/examples/auto/* auto/
```

340 Editar `auto/config`, afegint les opcions més adients. Per exemple:

```
#!/bin/sh
lb config noauto \
  --architectures i386 \
  --linux-flavours 686-pae \
  --binary-images hdd \
  --mirror-bootstrap http://ftp.ch.debian.org/debian/ \
  --mirror-binary http://ftp.ch.debian.org/debian/ \
  "${@}"
```

342 Ara, cada vegada que s'utilitzi `lb config`, `auto/config` restablirà la configuració basada en aquestes opcions. Quan es vulgui fer canvis, editar les opcions d'aquest fitxer en lloc de passar-les a `lb config`. Quan s'utilitza `lb clean`, `auto/clean` netejarà els fitxers de `config/*` juntament amb els altres productes de construcció. I, finalment, quan s'utilitza `lb build`, es crea un log de la construcció mitjançant `auto/build` anomenat `build.log`.

343 **Nota:** Aquí s'utilitza un paràmetre especial `noauto` per a suprimir un altra crida a `auto/config`, la qual cosa impedeix la recursivitat infinita. Assegurar-se de no eliminarlo accidentalment fent canvis. També, tenir cura de que quan es divideix l'ordre `lb config` a través de diverses línies per a facilitar la lectura, com es mostra en l'exemple anterior, no s'oblidi la barra invertida ( al final de cada línia que segueix a la següent.

## 344 6.2 Clonar una configuració publicada via Git

345 Utilitzar l'opció `lb config --config` per a clonar un repositori Git que contingui una configuració en viu. Si es vol basar la configuració en un repositori mantingut pel Live Systems Project, mirar el repositori a <http://live-systems.org/gitweb/> amb el nom `live-images` sota el títol `Packages`. Aquest repositori conté les configuracions per a les **imatges prefabricades**

346 Per exemple, per a construir una imatge de rescat, utilitzar el repositori `live-images` de la manera següent:

```
$ mkdir live-images && cd live-images
$ lb config --config git://live-systems.org/git/live-images.git
$ cd images/rescue
```

348 Editar `auto/config` i qualsevol altra cosa necessària dins l'arbre `config` per a satisfer les necessitats pròpies. Per exemple, per a fer les imatges prefabricades no oficials que contenen paquets de la secció `non-free` simplement s'afegeix `--archive-areas "main contrib non-free"`.

349 Si es desitja, es pot definir una drecera en la configuració de Git, afegint el següent a `${HOME}/.gitconfig`:

```
[url "git://live-systems.org/git/"]
  insteadOf = lso:
```

351 Això permet utilitzar `lso:` en qualsevol lloc on cal especificar la direcció d'un repositori git. També es pot omitir el sufix `.git`, i així, començar una nova imatge amb aquesta configuració és tan fàcil com:

```
$ lb config --config lso:live-images
```

353 Clonar tot el repositori `live-images` còpia les configuracions utilitzades per diverses imatges. Si es vol construir una imatge diferent després d'haver acabat amb la primera, canviar a un altre directori i un altre cop i, opcionalment, fer els canvis per a adaptar-les a les necessitats pròpies.

354 En qualsevol cas, recordar que cada vegada que s'ha de construir una imatge, s'ha de fer com a superusuari: `lb build`



## Personalització dels continguts

## 7. Visió general de la personalització

En aquest capítol s'ofereix una visió general de les diverses formes en què es pot personalitzar un sistema en viu.

### 7.1 Configuració durant la construcció vs. durant l'arrencada

La configuració de un sistema en viu es divideix en opcions en temps de construcció que són les opcions que s'apliquen durant la seva creació i les opcions d'arrencada del sistema que s'apliquen durant l'arrencada. Les opcions d'arrencada es divideixen en les què ocorren al principi de l'arrencada, aplicades pel paquet *live-boot*, i les que ocorren més tard en l'arrencada, aplicades per *live-config*. Qualsevol opció durant l'arrencada pot ser modificada per l'usuari, especificant-la a l'indicador d'arrencada. La imatge també pot ser construïda amb els paràmetres d'arrencada per defecte perquè els usuaris puguin simplement arrencar el sistema en viu sense especificar cap altra opció, ja que tots els valors per defecte són adequats. En particular, l'argument `lb --bootappend-live` consta de les opcions de línia d'ordres per defecte del nucli per al sistema en viu, com ara la persistència, la distribució del teclat o la zona horària. Veure «[Personalització de l'entorn local i el llenguatge](#)», per exemple.

Les opcions de configuració durant la construcció es descriuen a la pàgina del manual de `lb config`. Les opcions durant l'arrencada es descriuen a les pàgines del manual de *live-boot* i *live-config*. Malgrat que els paquets *live-boot* i *live-config* s'instal·len en el sistema en viu que s'està construint, és recomana instal·lar-los en el sistema de construcció per a tenir una referència fàcil quan s'està treballant en la configuració. És segur fer-ho, ja que cap dels

scripts continguts en ells s'executen a menys que el sistema s'hagi configurat com a sistema viu.

### 7.2 Etapes de la construcció

El procés de construcció es divideix en etapes, amb personalitzacionsS362 diferents aplicades successivament en cada una. La primera etapa que s'executa es la fase **bootstrap**. Aquesta és la fase inicial de poblar el directori chroot amb paquets per a fer un sistema Debian bàsic. Això és seguit per l'etapa **chroot**, que completa la construcció del directori chroot, omplint-lo amb tots els paquets que s'indiquen en la configuració, juntament amb qualsevol altre material. La majoria de personalitzacions dels continguts es produeixen en aquesta etapa. L'etapa final de preparació de la imatge en viu és l'etapa **binary**, quan es construeix una imatge capaç d'arrencar, amb el contingut del directori chroot per a construir el sistema de fitxers arrel per al sistema en viu, i que inclou el programa de instal·lació i qualsevol altre material addicional en el medi de destinació fora del sistema de fitxers del sistema en viu. Després de construir la imatge en viu, si està habilitat, s'inclou el codi font original a l'etapa **source**.

Dins de cadascuna d'aquestes etapes, hi ha una seqüència particular en la qual s'apliquen les ordres. Això es fa de manera que es garanteixi que les personalitzacions es poden superposar de manera raonable. Per exemple, dins l'etapa **chroot**, les preconfiguracions (preseeds) s'apliquen abans que s'instal·lin els paquets, els paquets s'instal·len abans que es copiïn els fitxers locals, i els ganxos s'executen més tard, després que tots els materials estiguin al seu lloc.

### 7.3 Suplementar lb config amb fitxers

364

365 Encara que lb config crea una configuració en esquelet al directori `config/`, per a aconseguir els objectius, pot ser necessari proporcionar fitxers addicionals en els subdirectoris de `config/`. Depenent d'on s'emmagatzemen els fitxers en la configuració, poden ser copiats en el sistema de fitxers del sistema en viu o en el sistema de fitxers de la imatge binària, o es pot proporcionar configuracions en temps de construcció del sistema que serien engorroses de passar com opcions de línia d'ordres. Es pot incloure coses com ara llistes personalitzades de paquets, art personalitzat o scripts ganxo per a ser executats ja sigui en temps de construcció o en temps d'arrencada, augmentant la flexibilitat ja considerable de debian-live amb codi propi.

### 7.4 Tasques de personalització

366

367 Els següents capítols s'organitzen pel tipus de tasques de personalització que els usuaris solen realitzar: **«Personalització de la instal·lació de paquets»**, **«Personalització dels continguts»** i **«Personalització de l'entorn local i el llenguatge»** cobreixen només algunes de les coses que es poden fer.

## Personalització de la instal·lació de paquets

## 8. Personalització de la instal·lació de paquets

La personalització més bàsica d'un sistema en viu pot ser la selecció dels paquets que seran inclosos en la imatge. Aquest capítol explica les diverses opcions de *live-build* per a personalitzar la instal·lació de paquets durant la construcció. Les opcions més importants que influeixen en els paquets que estan disponibles per a ser instal·lats en la imatge són les àrees de distribució i el arxiu. Per a garantir velocitats de descàrrega decents, s'ha de triar un mirall de distribució proper. També es pot incloure repositoris de backports, paquets experimentals o personalitzats, o incloure paquets directament com si fossin fitxers. Es poden definir llistes de paquets, incloent-hi els metapaquets que instal·laran diversos paquets relacionats alhora, com ara paquets per a un ordinador d'escriptori o un llenguatge en particular. Finalment, una sèrie d'opcions donen un cert control sobre *apt* o si es prefereix *aptitude*, quan s'instal·len els paquets durant la construcció. Això pot ser útil si s'utilitza un proxy, es vol desactivar la instal·lació de paquets recomanats per a estalviar espai, o hi ha la necessitat de controlar quines versions dels paquets s'instal·len mitjançant la tècnica pinning d'APT, per nomenar algunes possibilitats.

### 8.1 Fonts dels paquets

#### 8.1.1 Distribució, àrees d'arxiu i mode

La distribució que es tria té una gran importància en els paquets que estan disponibles per a incloure en una imatge en viu. Només cal especificar el nom en clau, que per defecte és **jessie** per a la versió **jessie** de *live-build*. Qualsevol distribució disponible a l'arxiu pot ser especificada pel seu nom en clau aquí. (Veure

<Termes> per a més detalls.) L'opció `--distribution` no només influeix en l'origen dels paquets dins l'arxiu, sinó que també instrueix a *live-build* per a comportar-se segons sigui necessari per a construir cada distribució suportada. Per exemple, per a construir la distribució **unstable**, **sid**, s'ha d'especificar:

```
$ lb config --distribution sid
```

A l'arxiu de la distribució, les àrees són les divisions principals de l'arxiu. A Debian, es tracta de `main`, `contrib` i `non-free`. Només `main` conté el programari que és part de la distribució Debian, per tant és el valor per defecte. Es poden especificar un o més valors, per exemple:

```
$ lb config --archive-areas "main contrib non-free"
```

Es dona suport experimental a alguns derivats de Debian a través de l'opció `--mode`. Per defecte, aquesta opció és `debian` però només si s'està construint en un sistema Debian o en un sistema desconegut. Si s'especifica amb `lb config` que es vol construir un dels derivats suportats alhora es modificaran les opcions per a crear aquest derivat. Si per exemple s'utilitza `lb config` amb el mode `ubuntu`, s'utilitzarà el nom de la distribució i les àrees dels arxius del derivat especificat en lloc dels de Debian. El «mode» també modifica el comportament de *live-build* per a adaptar-lo als derivats.

**Nota:** Els projectes per als quals s'han afegit aquests modes són els principals responsables de donar suport als usuaris d'aquestes opcions. El Live Systems Project, al seu torn, dona suport de desenvolupament només sobre una base de millor

esforç, basada en les informacions proporcionades pels projectes derivats ja que nosaltres no desenvolupem ni donem suport a aquests derivats.

### 8.1.2 Miralls de distribució

L'arxiu de Debian es replica a través d'una àmplia xarxa de miralls a tot el món perquè la gent de cada regió pugui triar un mirall proper amb la millor velocitat de descàrrega. Cadascuna de les opcions `--mirror-*` governa quin mirall de distribució s'utilitzarà en les diverses etapes de la construcció. Recordar de **«Etapas de la construcció»** que l'etapa **bootstrap** es quan el chroot s'omple inicialment per *debootstrap* amb un sistema mínim i l'etapa **chroot** és quan s'utilitza el chroot per a la construcció del sistema de fitxers del sistema en viu. D'aquesta manera, s'utilitzen els miralls corresponents per a aquestes etapes, i més tard, durant l'etapa **binary** s'utilitzen els valors `--mirror-binary` i `--mirror-binary-security` substituint qualsevol mirall utilitzat en una etapa anterior.

### 8.1.3 Miralls de distribució utilitzats en temps de construcció

Per a establir els miralls de la distribució utilitzats en temps de construcció perquè apuntin a una rèplica local, és suficient establir `--mirror-bootstrap`, `--mirror-chroot-security` i `--mirror-chroot-backports` de la manera següent.

```
$ lb config --mirror-bootstrap http://localhost/debian/ \
  --mirror-chroot-security http://localhost/debian-security/ \
  --mirror-chroot-backports http://localhost/debian-backports/
```

El mirall per al chroot, especificat per l'opció `--mirror-chroot`, per defecte pren el mateix valor que `--mirror-bootstrap`

### 8.1.4 Miralls de distribució utilitzats en temps d'execució

Les opcions `--mirror-binary*` governen els miralls de distribució que acaben a la imatge binària. Aquestes poden ser utilitzades per a instal·lar paquets addicionals mentre s'executa el sistema en viu. Els valors per defecte fan servir `http.debian.net`, un servei que tria un mirall geogràficament a prop en funció, entre altres coses, de la família de la IP de l'usuari i de la disponibilitat del mirall. Aquesta és una opció adequada quan no es pot predir quin serà el millor mirall per a tots els usuaris. O es pot especificar els valors propis com es mostra en l'exemple següent. Una imatge construïda a partir d'aquesta configuració només seria convenient per als usuaris en una xarxa on "mirror" és abastable.

```
$ lb config --mirror-binary http://mirror/debian/ \
  --mirror-binary-security http://mirror/debian-security/ \
  --mirror-binary-backports http://mirror/debian-backports/
```

### 8.1.5 Repositoris addicionals

És possible afegir més repositoris, ampliant les opcions de paquets més enllà dels disponibles en la pròpia distribució de destinació. Aquests poden ser, per exemple, per a backports, experimentals o paquets personalitzats. Per a configurar repositoris addicionals, crear els fitxers `config/archives/your-repository.list.chroot`, `i/o config/archives/your-repository.list.binary`. Igual que amb les opcions `--mirror-*` aquest regeix els repositoris utilitzats en l'etapa **chroot** durant la construcció de la imatge, i a l'etapa

**binary** , és a dir, per a ser utilitzades quan s'executa el sistema en viu.

Per exemple, `config/archives/live.list.chroot` permet instal·lar paquets des del repositori d'instantànies de `debian-live` en el moment de construcció del sistema viu.

```
deb http://live-systems.org/ sid-snapshots main contrib non-free
```

Si s'afegeix la mateixa línia a `config/archives/live.list.binary`, el repositori sera afegit al directori `/etc/apt/sources.list.d/` del sistema viu.

Si aquests fitxers existeixen, són utilitzats de forma automàtica.

També s'ha de posar la clau GPG utilitzada per a signar el repositori en fitxers `config/archives/your-repository.key.{binary,chroot}`.

En cas de necessitar un APT pinning personalitzat, es poden col·locar les preferències APT en fitxers `config/archives/your-repository.pref.{binary,chroot}`, que seran afegits automàticament al sistema en viu al directori `/etc/apt/preferences.d/`.

## 8.2 Selecció dels paquets a instal·lar

Hi ha una sèrie de formes de triar els paquets que *live-build* instal·larà en la imatge, que abasta una varietat de necessitats diferents. Es pot simplement anomenar paquets individualment per a instal·lar en una llista de paquets. També es pot optar per utilitzar metapaquets a les llistes, o seleccionar-los utilitzant camps de control de fitxers de paquets. I, finalment, es poden copiar paquets com si fossin fitxers dins del arbre `config/`, que és

un mètode que s'adapta perfectament a fer proves amb paquets nous o experimentals abans de afegirlos a un repositori.

### 8.2.1 Llistes de paquets

Les llistes de paquets són una forma eficaç d'expressar quins paquets han de ser instal·lats. La sintaxi de la llista suporta seccions condicionals que fa que sigui fàcil construir llistes i adaptar-les per al propi ús en múltiples configuracions. Els noms dels paquets també poden ser injectats a la llista amb ajudants de l'interpret d'ordres en temps de construcció.

**Nota:** El comportament de *live-build* a l'hora d'especificar un paquet que no existeix està determinat per la elecció que es faci de l'eina APT. Veure [«Elegir apt or aptitude»](#) per a més detalls.

### 8.2.2 Ús dels metapaquets

La forma més senzilla per a omplir la llista de paquets és utilitzar una tasca metapaquet mantinguda per una distribució. Per exemple:

```
$ lb config
$ echo task-gnome-desktop > config/package-lists/desktop.list.chroot
```

Això reemplaça l'antic mètode de llistes predefinides de *live-build* 2.x. A diferència de les llistes predefinides, els metapaquets no són específics del projecte Live Systems. Per contra, són mantinguts per grups d'especialistes que treballen dins la distribució i per tant, reflecteixen el consens de cada grup sobre els paquets que serviran millor a les necessitats dels usuaris. A més, abasten una gamma molt més àmplia de casos d'ús que les

l·listes predefinides que substitueixen.

Tots els metapaquets tenen el prefix `task-`, de manera que una forma ràpida de determinar quins estan disponibles (encara que pot contenir un grapat d'entrades falses que coincideixin amb el nom, però que no són metapaquets) és fer coincidir el nom del paquet amb:

```
$ apt-cache search --names-only ^task-
```

A més d'aquests, es troben altres metapaquets amb diverses finalitats. Alguns són subconjunts de paquets de tasques més àmplies, com `gnome-core`, mentre que altres són parts individuals especialitzades de un Debian Pure Blend, com els metapaquets `education-*`. Per a obtenir una llista de tots els metapaquets que hi ha a l'arxiu, instal·lar el paquet `debtags` i llistar tots els paquets amb l'etiqueta `role::metapackage` de la següent manera:

```
$ debtags search role::metapackage
```

### 8.2.3 Llistes locals de paquets

Ja sigui afegint metapaquets a una llista, paquets individuals, o una combinació d'ambdós, totes les llistes de paquets locals s'emmagatzemen a `config/package-lists/`. Es pot utilitzar més d'una llista i això es presta molt bé als dissenys modulars. Per exemple, es pot decidir dedicar una llista a una elecció particular d'escriptori, l'altra a una col·lecció de paquets relacionats que puguin ser fàcilment utilitzats al damunt d'un escriptori diferent. Això permet experimentar amb diferents combinacions de conjunts

de paquets amb un mínim d'esforç, intercanviant llistes comunes entre els diferents projectes d'imatges en viu.

Les llistes de paquets que es troben en aquest directori han de tenir el sufix `.list` per a ser processades, i a més a més un sufix d'etapa adicional `.chroot` o `.binary` per a indicar per a quina etapa és la llista.

**Nota:** Si no s'especifica el sufix d'etapa, la llista s'utilitzarà per a ambdues etapes. Normalment, s'especifica `.list.chroot` de manera que els paquets només s'instal·laran al sistema de fitxers en viu i no hi haurà una còpia extra del `.deb` en el medi.

### 8.2.4 Llistes locals de paquets per a l'etapa binary

Per a crear una llista per a l'etapa binary, crear un fitxer amb el sufix `.list.binary` a `config/package-lists/`. Aquests paquets no s'instal·len al sistema de fitxers en viu però s'inclouen en el medi en viu al directori `pool/`. Un ús típic d'aquesta llista seria amb una de les variants del instal·lador `non-live`. Com s'ha esmentat anteriorment, si es vol que aquesta llista sigui la mateixa que la llista de l'etapa `chroot`, simplement utilitzar el sufix `.list`.

### 8.2.5 Generar llistes de paquets

De vegades passa que la millor manera de crear una llista és generar-la amb un script. Qualsevol línia que comença amb un signe d'exclamació indica una ordre que s'executarà dins del `chroot` quan la imatge es construeix. Per exemple, es podria incloure la línia `! grep-aptavail -n -sPackage -FPriority standard | sort` en una llista de paquets per a produir una llista ordenada de paquets disponibles amb `Priority: standard`.

De fet, la selecció de paquets amb l'ordre `grep-aptavail` (del

paquet `dctrl-tools`) és tan útil que `live-build` proporciona un script `Packages` d'ajuda per motius de comoditat. Aquest script accepta dos arguments: `field` i `pattern`. Per tant, es pot crear una llista amb els següents continguts:

```
$ lb config
$ echo '! Packages Priority standard' > config/package-lists/standard.list.<↵
chroot
```

## 8.2.6 Ús de condicionals dins de les llistes de paquets

Qualsevol de les variables de configuració de `live-build` emmagatzemades a `config/*` (menys el prefix `LB_`) poden ser utilitzades en sentències condicionals en les llistes de paquets. En general, això significa qualsevol opció `lb config` en lletres majúscules i amb guions canviats a guions baixos. Però a la pràctica, només tenen sentit les que influeixen en la selecció de paquets, com ara `DISTRIBUTION`, `ARCHITECTURES` o `ARCHIVE_AREAS`.

Per exemple, per a instal·lar `ia32-libs` si s'especifica `--architectures amd64`:

```
#if ARCHITECTURES amd64
ia32-libs
#endif
```

És possible fer un test d'un nombre de valors, per exemple per a instal·lar `memtest86+` si s'especifica `--architectures i386` o `--architectures amd64`:

```
#if ARCHITECTURES i386 amd64
memtest86+
#endif
```

També es pot provar amb variables que poden contenir més d'un valor, per exemple, per a instal·lar `vrms` si s'especifica `contrib` o `non-free` a través de l'opció `--archive-areas`:

```
#if ARCHIVE_AREAS contrib non-free
vrms
#endif
```

No és possible el anidament dels condicionals.

## 8.2.7 Eliminar paquets durant la instal·lació

Es pot crear llistes de paquets en fitxers amb els sufixos `.list.chroot_live` i `.list.chroot_install` dins del directori `config/package-lists`. Si hi ha una llista «live» i una llista «install» els paquets de la llista `.list.chroot_live` s'eliminaran amb un script ganxo després de la instal·lació (si l'usuari utilitza l'instal·lador). Els paquets de la llista `.list.chroot_install` seran presents tant en el sistema en viu com en el sistema instal·lat. Aquest és un cas especial per al programa d'instal·lació i pot ser útil si es té `--debian-installer live` establert en la configuració i es desitja eliminar paquets específics del sistema en viu durant la instal·lació.

## 8.2.8 Tasques d'escriptori i llenguatge

Les tasques d'escriptori i el llenguatge són casos especials que necessiten una mica de planificació i configuració extra. Les

imatges en viu són diferents de les imatges de l'instal·lador de Debian en aquest sentit. A l'instal·lador de Debian, si el medi es va preparar per a obtenir un tipus d'entorn d'escriptori en particular, la tasca corresponent s'instal·larà automàticament. Per tant hi ha tasques internes `gnome-desktop`, `kde-desktop`, `lxde-desktop` i `xfce-desktop`, cap de les quals s'ofereixen al menú de `tasksel`. De la mateixa manera, no hi ha cap entrada de menú per a tasques de llengües, però l'elecció del idioma de l'usuari durant la instal·lació influeix en la selecció de les tasques de les llengües corresponents.

432 En el desenvolupament d'una imatge en viu d'escriptori, la imatge sol arrencar directament a un escriptori de treball, les opcions d'escriptori i de llengua han estat fetes en temps de construcció, no en temps d'execució com en el cas del instal·lador de Debian. Això no vol dir que una imatge en viu no es pugui construir per a donar suport a diversos equips d'escriptori o diversos idiomes i oferir a l'usuari una opció, però això no és el comportament de *live-build* per defecte.

433 Com que no hi ha cap ajust automàtic per a les tasques de llengua que incloguin coses com ara tipus de lletres específics per a una llengua o paquets de mètode d'entrada, si es vol, cal especificar-ho en la configuració. Per exemple, una imatge d'escriptori GNOME que contingui suport per al alemany podria incloure les següents tasques metapaquets:

434

```
$ lb config
$ echo "task-gnome-desktop task-laptop" >> config/package-lists/my.list.<↵
  chroot
$ echo "task-german task-german-desktop task-german-gnome-desktop" >> <↵
  config/package-lists/my.list.chroot
```

## 8.2.9 Tipus i versió del nucli

435

Depenent de l'arquitectura, s'inclouran per defecte en la imatge un o més tipus de nuclis. Es pot triar diferents tipus a través de l'opció `--linux-flavours`. Cada tipus té un sufix per a l'arrel per defecte `linux-image` per a formar el nom de cada metapaquet que al seu torn depèn d'un paquet del nucli exacte que s'ha d'incloure en la imatge.

436

Així, per defecte, una imatge per a l'arquitectura `amd64` inclourà el metapaquet `linux-image-amd64` i una imatge per a l'arquitectura `i386` inclourà els metapaquets `linux-image-486` i `linux-image-686-pae`. Al moment d'escriure, aquests paquets depenen de `linux-image-3.2.0-4-amd64`, `linux-image-3.2.0-4-486` i `linux-image-3.2.0-4-686-pae`, respectivament.

437

Quan hi ha més d'una versió del nucli disponible en els arxius configurats, es pot especificar el nom d'un paquet del nucli amb l'opció `--linux-packages`. Per exemple, suposem que s'està construint una imatge d'arquitectura `amd64` i es vol afegir l'arxiu experimental amb propòsits de fer proves perquè es pugui instal·lar el nucli `linux-image-3.7-trunk-amd64`. Es podria configurar la imatge de la següent manera:

438

```
$ lb config --linux-packages linux-image-3.7-trunk
$ echo "deb http://ftp.debian.org/debian/ experimental main" > config/<↵
  archives/experimental.list.chroot
```

439

## 8.2.10 Nuclis personalitzats

440

Es pot construir i incloure nuclis propis personalitzats, sempre que s'integrin en el sistema de gestió de paquets de Debian. El

441



sistema de *live-build* no és compatible amb nuclis no construïts com paquets `.deb`.

La manera apropiada i recomanable d'implementar els propis paquets del nucli és seguir les instruccions del `kernel-handbook`. Recordar que s'ha de modificar l'ABI i els sufixos del tipus apropiadament, i a continuació, incloure un conjunt complet dels `packages` que corresponen amb `linux` i `linux-latest` al repositori.

Si s'opta per construir els paquets del nucli sense els metapaquets a joc, cal especificar una arrel `--linux-packages` apropiada com s'indica a **Tipus i versió del nucli**. Com expliquem a **Instal·lació de paquets modificats o de tercers**, és millor si s'inclouen els paquets del nucli personalitzat en un repositori propi, tot i que les alternatives discutides en aquella secció també funcionen.

Està més enllà de l'abast d'aquest document donar consells sobre com personalitzar un nucli. No obstant això, cal, almenys, assegurar-se que la configuració compleix els següents requisits mínims:

- Utilitzar una `ramdisk` inicial.
- Incloure el mòdul d'unió del sistema de fitxers (normalment `aufs`).
- Incloure tots els mòduls del sistema d'arxius requerits per la configuració (normalment `squashfs`).

### 8.3 Instal·lació de paquets modificats o de tercers

Si bé està en contra de la filosofia d'un sistema en viu, de vegades pot ser necessària la construcció d'un sistema amb versions modificades dels paquets que es troben al arxiu de Debian. Pot ser per a modificar o donar suport a funcions addicionals, les llengües o les marques, o fins i tot per a eliminar elements dels paquets existents que són indesitjables. De la mateixa manera, es

poden utilitzar paquets de tercers per a afegir alguna funcionalitat personalitzada i/o propietària.

Aquesta secció no cobreix l'assessorament en matèria de construcció o manteniment de paquets modificats. Però el mètode de Joachim Breitner's 'How to fork privately' a <http://www.joachim-breitner.de/blog/archives/282-How-to-fork-privately.html> pot ser d'interès. La creació de paquets personalitzats es tracta a Debian New Maintainers' Guide at <https://www.debian.org/doc/maint-guide/> i en altres llocs.

Hi ha dues formes d'instal·lar paquets personalitzats modificats:

- `packages.chroot`
- L'ús d'un repositori APT personalitzat

Utilitzar `packages.chroot` és més fàcil d'aconseguir i útil per a personalitzacions "ràpides", però té una sèrie d'inconvenients, mentre que l'ús d'un repositori APT personalitzat és més costós en la quantitat de temps necessari per a posar-lo en marxa.

#### 8.3.1 Fer servir `packages.chroot` per a instal·lar paquets personalitzats

Per a instal·lar un paquet personalitzat, només s'ha de copiar al directori `config/packages.chroot/`. Els paquets que es troben dins d'aquest directori s'instal·laran automàticament en el sistema en viu durant la construcció - no cal especificar res més en cap altre lloc.

Els paquets **han de** ser nomenats en la forma prescrita. Una manera simple de fer això és utilitzar `dpkg-name`.

Utilitzar `packages.chroot` per a la instal·lació de paquets personalitzats té els seus desavantatges:

- No és possible utilitzar APT segur.

- Cal posar tots els paquets apropiats al directori `config/packages.chroot/`.
- No es adient per a l'emmagatzematge de configuracions de sistemes en viu en el control de revisió.

### 8.3.2 Fer servir un repositori APT per a instal·lar paquets personalitzats

A diferència de `packages.chroot`, quan s'utilitza un repositori APT personalitzat s'ha d'assegurar que s'especifiquen els paquets en un altre lloc. Veure [«Selecció dels paquets a instal·lar»](#) per a més detalls.

Si bé crear un repositori APT per a instal·lar paquets personalitzats pot semblar un esforç innecessari, la infraestructura pot ser fàcilment reutilitzada en una data posterior per a oferir actualitzacions dels paquets modificats.

### 8.3.3 Paquets personalitzats i APT

*live-build* utilitza APT per a instal·lar tots els paquets al sistema en viu, per tant, heretarà els comportaments d'aquest programa. Un exemple rellevant és que (assumint una configuració per defecte) si es dóna el cas que un paquet està disponible en dos repositoris diferents, amb diferents números de versió, APT triarà per a instal·lar el paquet amb la versió més alta.

A causa d'això, s'aconsella augmentar el nombre de la versió dels paquets personalitzats als fixers `debian/changelog` per a assegurar-se que la versió modificada és la que s'instal·la en lloc d'una dels repositoris oficials de Debian. Això també es pot aconseguir mitjançant l'alteració de les preferències d'APT del sistema en viu - veure [«APT pinning»](#) per a més informació.

## 8.4 Configurar APT en temps de construcció

Es pot configurar APT a través d'una sèrie d'opcions que només s'apliquen en temps de construcció. (La configuració d'APT al sistema en funcionament en viu es pot fer de forma normal per als continguts del sistema en viu, és a dir, mitjançant la inclusió de les configuracions adequades a través de `config/includes.chroot/`.) Per a obtenir una llista completa, buscar les opcions que comencen amb `apt` a la pàgina del manual de `lb_config`.

### 8.4.1 Elegir apt o aptitude

Es pot optar per utilitzar *apt* o *aptitude* a l'hora d'instal·lar paquets en temps de construcció. Quina utilitat s'usa es configura gràcies al argument `--apt` de `lb_config`. Escollir el mètode d'implementació per al comportament preferit durant la instal·lació de paquets, la diferència notable és la forma en que es manegen els paquets que falten.

- `apt`: Amb aquest mètode, si un paquet que s'especifica falta, l'instal·lació de paquets fallarà. Aquesta és la configuració per defecte.
- `aptitude`: Amb aquest mètode, si s'especifica un paquet que falta, l'instal·lació de paquets tindrà èxit.

### 8.4.2 L'ús d'un proxy amb APT

Una configuració típica d'APT és per a fer front a la construcció d'una imatge darrere d'un proxy. Es pot especificar el proxy per a APT amb les opcions `--apt-ftp-proxy 0` o `--apt-http-proxy` segons sigui necessari, per exemple,

```
$ lb config --apt-http-proxy http://proxy/
```

### 8.4.3 Afinar APT per a estalviar espai

Pot ser necessari estalviar espai en el medi destinat a la imatge, en aquest cas una o altra o ambdós de les següents opcions poden ser d'interès.

Si no es vol incloure els índexs d'APT en la imatge, es poden omitir amb:

```
$ lb config --apt-indices false
```

Això no influirà en les entrades de `/etc/apt/sources.list`, sinó simplement si `/var/lib/apt` conté els fitxers dels índexs o no. El desavantatge és que APT necessita aquests índexs per tal d'operar en el sistema en viu, així que abans d'executar per exemple `apt-cache search` o `apt-get install`, l'usuari primer ha fer un `apt-get update` per a crear aquests índexs.

Si es considera que la instal·lació de tots els paquets recomanats infla massa la imatge, sempre que s'estigui preparat per a fer front a les conseqüències que s'analitzen a continuació, es pot desactivar aquesta opció per defecte d'APT amb:

```
$ lb config --apt-recommends false
```

La conseqüència més important de desactivar els «recommends» és que `live-boot` i `live-config` recomanen alguns paquets que proporcionen una funcionalitat important utilitzada per la majoria

de configuracions Live, com per exemple `user-setup` recomanat per `live-config` i que s'utilitza per a crear l'usuari en viu. En tots menys en els casos més excepcionals es necessita tornar a afegir almenys alguns dels recommends a les llistes de paquets o en cas contrari la imatge no funcionarà com s'espera, si és que funciona. Mirar els paquets recomanats per a cada un dels paquets inclosos en la construcció i si no s'està segur que es poden ometre, tornar a afegir-los a les llistes de paquets.

La conseqüència més general aquí és que si no s'instal·len els paquets recomanats per un paquet determinat, és a dir, "els paquets que es troben junts amb aquest en totes les instal·lacions a menys que siguin inusuals" (Debian Policy Manual, secció 7.2), alguns paquets que en realitat són necessaris per als usuaris del sistema Live poden ser omesos. Per tant, suggerim revisar la diferència que desactivar els paquets recomanats té en la llista de paquets (veure el fitxer `binary.packages` generat per `lb build`) i tornar a incloure a la llista els paquets que falten que haurien de ser instal·lats. Si només es vol deixar de banda un petit nombre de paquets recomanats, es pot deixar els «recommends» activats i establir una prioritat pin d'APT negativa en els paquets seleccionats per a impedir la seva instal·lació, com s'explica a [«APT pinning»](#).

### 8.4.4 Passar opcions per a apt o aptitude

Si no hi ha cap opció `lb config` per a modificar el comportament d'APT de la manera què es necessita, es pot utilitzar `--apt-options` o `--aptitude-options` per a passar opcions a l'eina APT configurada. Consultar les pàgines dels manual `apt` i `aptitude` per a més detalls. Tenir en compte que ambdues opcions tenen valors per defecte que s'hauran de mantenir, a més de les opcions que es proporcionen. Així, per exemple, suposant que s'ha inclòs algun paquet de `snapshot.debian.org` per a fer proves i es vol

especificar `Acquire::Check-Valid-Until=false` perquè APT no es queixi de que el fitxer `Release` ja ha caducat es podria fer com en el exemple següent, afegint la nova opció després del valor per defecte `--yes`:

```
$ lb config --apt-options "--yes -oAcquire::Check-Valid-Until=false"
```

Consultar les pàgines del manual per a entendre completament aquestes opcions i quan utilitzar-les. Això és només un exemple i no s'ha d'interpretar com un consell per a configurar la imatge. Aquesta opció no seria adequada, per exemple, per a una versió final d'una imatge en viu.

Per a configuracions més complicades que impliquen opcions `apt.conf` pot ser adequat crear un fitxer `config/apt/apt.conf`. Consultar també les altres opcions `apt-*` per a tenir algunes dreceres convenients per a les opcions que es necessiten amb freqüència.

## 8.4.5 APT pinning

Com a referència, llegir primer la pàgina del manual `apt_preferences(5)`. Es pot configurar APT pinning durant la construcció, o bé durant l'execució. En el primer cas, crear `config/archives/*.pref`, `config/archives/*.pref.chroot`, i `config/apt/preferences`. Per al segon cas, crear `config/includes.chroot/etc/apt/preferences`.

Suposem que s'està construint un sistema en viu **jessie** però es necessita que tots els paquets «live-» que acaben dins de la imatge binària s'instal·lin desde **sid** en temps de construcció. Cal afegir **sid** a les fonts d'APT i fer un pin dels paquets live amb una prioritat més alta, però tots els altres paquets amb una prioritat més baixa

que la prioritat per defecte de manera que només els paquets que es vol s'instal·lin desde **sid** en el moment de la construcció i tots els altres es prenguin de la distribució de destinació, **jessie**. Això es pot aconseguir de la manera següent:

```
$ echo "deb http://mirror/debian/ sid main" > config/archives/sid.list.<↵
chroot
$ cat >> config/archives/sid.pref.chroot << EOF
Package: live-*
Pin: release n=sid
Pin-Priority: 600

Package: *
Pin: release n=sid
Pin-Priority: 1
EOF
```

Una prioritat pin negativa evitarà que un paquet s'instal·li, com en el cas que no es vulgui un paquet que és recomanat per un altre paquet. Suposem que s'està construint una imatge LXDE afegint `task-lxde-desktop` a `config/package-lists/desktop-list.chroot` però no es desitja que al usuari se li demani que guardi les contrasenyes wifi al keyring. Aquesta llista depèn de *lxde-core*, que recomana *gksu*, que al seu torn recomana *gnome-keyring*. Si es vol omitir el paquet recomanat *gnome-keyring*, es pot fer mitjançant l'addició de les següents línies a `config/apt/preferences`:

```
Package: gnome-keyring
Pin: version *
Pin-Priority: -1
```

## Personalització dels continguts

## 9. Personalització dels continguts

Aquest capítol tracta d'afinar la personalització dels continguts del sistema en viu més enllà de simplement triar els paquets que es desitja incloure. Els «includes» permeten afegir o reemplaçar fitxers arbitraris en la imatge en viu, els scripts ganxo (hooks) permeten executar ordres arbitràries en diferents etapes de la construcció i en el moment d'arrencar, i la preconfiguració (preseeding) permet configurar els paquets quan s'instal·len proporcionant respostes a les preguntes de debconf.

### 9.1 Includes

Tot i que l'ideal seria un sistema en viu que inclogués només fitxers proporcionats per paquets Debian sense modificació, de vegades és convenient proporcionar o modificar part del contingut a través de fitxers. Amb els includes, és possible afegir (o substituir) fitxers arbitraris en la imatge en viu. *live-build* ofereix dos mecanismes per al seu ús:

- Chroot local includes: Aquests permeten afegir o substituir fitxers dintre de chroot/Live en el sistema de fitxers. Consultar [«Live/chroot local includes»](#) per a més informació.
- Binary local includes: Aquests permeten afegir o substituir fitxers dins la imatge binària. Consultar [«Binary local includes»](#) per a més informació.

Consultar [«Termes»](#) per a més informació sobre la distinció entre les imatges “Live” and “binary”.

#### 9.1.1 Live/chroot local includes

Es poden utilitzar els chroot local includes per a afegir o reemplaçar fitxers en el sistema de fitxers chroot/Live perquè puguin ser utilitzats en el sistema en viu. Un ús típic és per a omplir l'esquelet del directori de l'usuari (/etc/skel) utilitzat pel sistema en viu per a crear el directori home de l'usuari en viu. Un altre és el de subministrar fitxers de configuració que poden ser simplement afegits o reemplaçats en la imatge sense processar; veure [«Live/chroot local hooks»](#) si es necessita processar-los.

Per a incloure fitxers, només s'han d'afegir al directori config/includes.chroot. Aquest directori es correspon amb el directori arrel / del sistema en viu. Per exemple, per a afegir un fitxer /var/www/index.html en el sistema en viu, fer:

```
$ mkdir -p config/includes.chroot/var/www
$ cp /path/to/my/index.html config/includes.chroot/var/www
```

La configuració tindrà llavors l'estructura següent:

```
-- config
[... ]
|-- includes.chroot
|   |-- var
|       |-- www
|           |-- index.html
[... ]
```

Els chroot local includes s'instal·len després de la instal·lació del paquets de tal manera que es sobreescrueixen els fitxers instal·lats pels paquets.

### 9.1.2 Binary local includes

Per a incloure material com documentació o vídeos en el sistema de fitxers del medi en viu de manera que sigui accessible immediatament després de la inserció del medi sense haver de arrencar el sistema en viu, es pot utilitzar els binary local includes. Això funciona de manera similar als chroot local includes. Per exemple, si els fitxers `~/video_demo.*` són vídeos de demostració del sistema en viu descrits i lligats per una pàgina d'índex HTML. Només cal copiar el material a `config/includes.binary/` de la següent manera:

```
$ cp ~/video_demo.* config/includes.binary/
```

Aquests fitxers apareixeran ara en el directori arrel del medi en viu.

## 9.2 Scripts ganxo (Hooks)

Els scripts ganxo permeten executar ordres en les etapes de la construcció chroot i binary per tal de personalitzar la imatge.

### 9.2.1 Live/chroot local hooks

Per a executar ordres durant l'etapa chroot, crear un script ganxo que contingui les ordres amb el sufix `.hook.chroot` i afegir-lo al directori `config/hooks/`. El ganxo s'executarà en el chroot després que la resta de la configuració del chroot s'hagi aplicat, assegurar-se que la configuració inclou tots els paquets i els fitxers que el ganxo necessita per funcionar. Veure els scripts chroot d'exemple per a diverses tasques comunes de personalització que

es poden trobar a `/usr/share/doc/live-build/examples/hooks` que es poden copiar o fer un enllaç simbòlic per a utilitzar-los en la pròpia configuració.

### 9.2.2 Scripts ganxo durant l'arrencada

Per a executar ordres durant l'arrencada, es pot proporcionar scripts ganxo per a *live-config* com s'explica a la secció "Personalització" de la seva pàgina del manual. Es poden afegir els ganxos de *live-config* a `/lib/live/config/`, tenint en compte la seqüència dels números. A continuació, afegir el script ganxo propi amb un número de seqüència apropiat com a prefix, ja sigui com a un chroot local include a `config/includes.chroot/-lib/live/config/`, o com un paquet personalitzat com es va discutir a [«Instal·lació de paquets modificats o de tercers»](#).

### 9.2.3 Binary local hooks

Per a executar ordres durant l'etapa binary, crear un script ganxo que contingui les ordres amb un sufix `.hook.binary` i afegir-lo al directori `config/hooks/`. El ganxo s'executarà després que s'executin totes les ordres de l'etapa binary però abans dels `binary_checksums`, la darrera ordre de l'etapa binary. Les ordres del ganxo no s'executen al chroot, per tant tenir cura de no modificar cap fitxer de fora del arbre de construcció, o es pot fer malbé el sistema de construcció! Veure els scripts ganxo binary d'exemple per a diverses tasques comunes de personalització a `/usr/share/doc/live-build/examples/hooks` que es poden copiar o fer un enllaç simbòlic per a utilitzar-los en la pròpia configuració.

### 9.3 Preconfiguració de les preguntes de Debconf

524

525 Els fitxers del directory `config/preseed/` amb el sufix `.cfg` seguits del sufix de l'etapa (`.chroot` o `.binary`) son considerats fitxers de preconfiguració de debconf i són instal·lats per *live-build* utilitzant `debconf-set-selections` durant l'etapa corresponent.

526 Per a més informació sobre debconf, veure `debconf(7)` del paquet *debconf*.

## Personalització dels comportaments en temps d'execució

### 10. Personalització dels comportaments en temps d'execució

Tota la configuració que es fa durant l'execució es feta per *live-config*. Aquestes són algunes de les opcions més comunes de *live-config* en que els usuaris estan interessats. Una llista completa de totes les possibilitats es poden trobar a la pàgina del manual de *live-config*.

#### 10.1 Personalitzar l'usuari en viu

Una consideració important és que l'usuari en viu es creat per *live-boot* durant l'arrencada i no per *live-build* en temps de construcció. Això influeix no només en on s'han de introduir els materials relacionats amb l'usuari durant la construcció, tal i com es va explicar a [«Live/chroot local includes»](#), sinó també en els grups i els permisos associats amb l'usuari.

Es poden especificar grups addicionals als que pertanyerà l'usuari en viu mitjançant l'ús de qualsevol de les possibilitats de configuració de *live-config*. Per exemple, per a afegir l'usuari en viu al grup fuse, es pot afegir el següent fitxer a `config/includes.chroot/etc/live/config/user-setup.conf`:

```
LIVE_USER_DEFAULT_GROUPS="audio cdrom dip floppy video plugdev netdev ↵
powerdev scanner bluetooth fuse"
```

o utilitzar `live-config.user-default-groups=audio,cdrom,dip,floppy,video,plugdev,netdev,powerdev,scanner,bluetooth,fuse` com paràmetre d'arrencada.

També és possible canviar el nom de l'usuari per defecte "user" i la contrasenya per defecte "live". Si es vol fer això per alguna raó, es pot aconseguir fàcilment de la següent manera:

Per a canviar el nom de l'usuari per defecte només s'ha d'especificar en la configuració:

```
$ lb config --bootappend-live "boot=live components username=live-user"
```

Una forma possible de canviar la contrasenya per defecte és per mitjà d'un ganxo com s'explica a [«Scripts ganxo durant l'arrencada»](#). Per a fer això, es pot utilitzar el script ganxo "passwd" de `/usr/share/doc/live-config/examples/hooks`, posar-li un prefix adequat (per exemple 2000-passwd) i afegir-lo a `config/includes.chroot/lib/live/config/`

#### 10.2 Personalització de l'entorn local i el llenguatge

Quan el sistema en viu arrenca, el llenguatge està implicat en dos passos:

- la generació de locales
- establir la configuració del teclat

La configuració local per defecte en la construcció d'un sistema viu és `locales=en_US.UTF-8`. Per a definir la locale que s'ha de generar, utilitzar el paràmetre `locales` de la opció `--bootappend-live` de `lb config`, per exemple.

```
$ lb config --bootappend-live "boot=live components locales=de_CH.UTF-8"
```



Es poden especificar diverses locales en una llista separada per comes.

Aquest paràmetre, així com els paràmetres de configuració del teclat que s'indiquen a continuació, també es pot utilitzar en la línia d'ordres del nucli. Es pot especificar una configuració regional mitjançant `language_country` (en aquest cas s'utilitza la codificació per defecte) o la forma completa `language_country.encoding`. Una llista de locales suportades i la codificació per a cadascuna es poden trobar a `/usr/share/i18n/SUPPORTED`.

`live-config` s'encarrega de la configuració del teclat per a X i per a la consola utilitzant el paquet `console-setup`. Per a la seva configuració es pot fer servir els paràmetres d'arrencada `keyboard-layouts`, `keyboard-variants`, `keyboard-options` i `keyboard-model` mitjançant l'opció `--bootappend-live`. Es poden trobar opcions vàlides per a aquests a `/usr/share/X11/xkb/rules/base.lst`. Per a trobar distribucions de teclat i variants per a un idioma determinat, s'ha d'intentar cercar el nom en anglès de la llengua i/o el país on es parla l'idioma, per exemple:

```
$ egrep -i '(!|german.*switzerland)' /usr/share/X11/xkb/rules/base.lst
! model
! layout
  ch                German (Switzerland)
! variant
  legacy            ch: German (Switzerland, legacy)
  de_noadkeys       ch: German (Switzerland, eliminate dead keys)
  de_sundeadkeys     ch: German (Switzerland, Sun dead keys)
  de_mac            ch: German (Switzerland, Macintosh)
! option
```

Tinir en compte que cada variant mostra la distribució que s'aplica en la descripció.

Sovint, només la distribució necessita ser configurada. Per

exemple, per a obtenir els fitxers de configuració regional per a la distribució del teclat alemany i suís-alemany per a l'entorn gràfic X:

```
$ lb config --bootappend-live "boot=live components locales=de_CH.UTF-8 ↵
keyboard-layouts=ch"
```

No obstant això, per als casos d'ús molt específics, potser es vol incloure altres paràmetres. Per exemple, per a establir un sistema francès, amb una distribució de teclat French-Dvorak (anomenat Bepo) en un teclat USB TypeMatrix EZ-Reach 2030, utilitzar:

```
$ lb config --bootappend-live \
  "boot=live components locales=fr_FR.UTF-8 keyboard-layouts=fr keyboard↵
  -variants=bepo keyboard-model=tm2030usb"
```

Es poden especificar diversos valors per a cada una de les opcions `keyboard-*` en una llista separada per comes amb l'excepció de `keyboard-model`, que només accepta un valor. Veure la pàgina del manual `keyboard(5)` per a més detalls i exemples de les variables `XKBMODEL`, `XKBLAYOUT`, `XKBVARIANT` i `XKBOPTIONS`. Si s'especifiquen diversos valors de `keyboard-variants` es correspondran un a un amb els valors `keyboard-layouts` (veure `setxkbmap(1)` opció `-variant`). Es poden utilitzar valors buits, per exemple, per a definir dos dissenys, el valor predeterminat `US QWERTY` i l'altre `US Dvorak`:

```
$ lb config --bootappend-live \
  "boot=live components keyboard-layouts=us,us keyboard-variants=,dvorak↵
  "
```

### 10.3 Persistència

Un paradigma d'un live cd és ser un sistema pre-instal·lat, que arrenca desde medis de només lectura, com un cdrom, on les modificacions no sobreviuen als reinicis del maquinari que l'executa.

Un sistema en viu és una generalització d'aquest paradigma i per tant, compatible amb altres medis, a més dels CDs, però tot i així, en el seu comportament per defecte, s'ha de considerar de només lectura i totes les evolucions en temps d'execució del sistema es perden al apagar l'equip.

La "Persistència" és un nom comú per a nomenar els diferents tipus de solucions per a guardar després de reiniciar algunes, o totes, les dades d'aquesta evolució en temps d'execució del sistema. Per a entendre com funciona, seria útil saber que, encara que el sistema s'inicia i s'executa des de medis de només lectura, les modificacions als fitxers i directoris s'escriuen ens medis d'escriptura, en general un ramdisk (tmpfs) i les dades dels discos ram no sobreviuen als reinicis.

Les dades emmagatzemades en aquest disc ram han de ser guardades en un suport d'escriptura persistent com medis d'emmagatzematge locals, un recurs compartit de xarxa o fins i tot una sessió d'una multisessió de un CD/DVD (re)grabable. Tots aquests medis són compatibles amb el sistema en viu de diferents maneres, i tots menys l'últim, requereixen un paràmetre especial que s'especifica en l'arrencada: `persistence`.

Si s'utilitza el paràmetre d'arrencada `persistence` (i no s'utilitza `nopersistence`) es proven els medis locals d'emmagatzematge (per exemple, discs durs, unitats USB) buscant volums amb persistència durant l'arrencada. És possible restringir els tipus de volums amb persistència que s'utilitzarà mitjançant l'especificació de certs paràmetres d'arrencada que es descriuen a la pàgina del

manual de *live-boot*(7). Un volum amb persistència és qualsevol dels següents:

- una partició, identificada pel seu nom GPT. 562
- un sistema de fitxers, identificat per la seva etiqueta de sistema de fitxers. 563
- un fitxer imatge situat en l'arrel de qualsevol sistema de fitxers llegibles (fins i tot una partició NTFS d'un altre SO), identificat pel seu nom de fitxer. 564

L'etiqueta de volum per als overlays ha de ser `persistence` però serà passat per alt a menys que contingui un fitxer anomenat `persistence.conf` que s'utilitza per a personalitzar completament la persistència del volum, és a dir, especificar els directoris que es volen conservar en el volum de persistència després de reiniciar. Veure [«El fitxer persistence.conf»](#) per a més detalls. 565

Aquests són alguns exemples de com preparar un volum que s'utilitzarà per a la persistència. Pot ser, per exemple, una partició ext4 en un disc dur o en una clau USB creat amb, per exemple: 566

```
# mkfs.ext4 -L persistence /dev/sdb1
```

Veure també [«Utilitzar l'espai lliure en una memòria USB»](#). 568

Si ja hi ha una partició al dispositiu, és pot canviar l'etiqueta amb un dels següents: 569

```
# tune2fs -L persistence /dev/sdb1 # for ext2,3,4 filesystems
```

Heus aquí un exemple de com crear un fitxer imatge basat en ext4 570

per a ser utilitzat per a la persistència:

```
$ dd if=/dev/null of=persistence bs=1 count=0 seek=1G # for a 1GB sized ↔
  image file
$ /sbin/mkfs.ext4 -F persistence
```

Un cop s'ha creat el fitxer imatge, per exemple, per a fer `/usr` persistent però només guardant els canvis que es fan en aquest directori i no tots els continguts de `/usr`, es pot utilitzar l'opció “union”. Si el fitxer imatge es troba en el directori home, copiar-lo a l'arrel del sistema de fitxers del disc dur i muntar-lo a `/mnt` de la següent manera:

```
# cp persistence /
# mount -t ext4 /persistence /mnt
```

A continuació, crear el fitxer `persistence.conf` afegint contingut i desmuntar el fitxer imatge.

```
# echo "/usr union" >> /mnt/persistence.conf
# umount /mnt
```

Ara, reiniciar el sistema i arrencar el medi en viu amb el paràmetre d'arrencada “persistence”.

### 10.3.1 El fitxer `persistence.conf`

Un volum amb l'etiqueta `persistence` ha de ser configurat mitjançant un fitxer `persistence.conf` per a fer directoris arbitraris persistents. Aquest fitxer, ubicat a l'arrel del sistema de fitxers

del volum, controla els directoris que fa persistents, i de quina manera.

A la pàgina del manual de `persistence.conf`(5) s'explica en detall com es configuren els muntatges de les overlays, però un simple exemple hauria de ser suficient per a la majoria d'usos. Si es vol fer el directori home i el directori del cache d'APT persistents en un sistema de fitxers ext4 a la partició `/dev/sdb1`:

```
# mkfs.ext4 -L persistence /dev/sdb1
# mount -t ext4 /dev/sdb1 /mnt
# echo "/home" >> /mnt/persistence.conf
# echo "/var/cache/apt" >> /mnt/persistence.conf
# umount /mnt
```

Després es reinicia el sistema. Durant la primera arrencada els continguts de `/home` i `/var/cache/apt` es copiaran en el volum de la persistència, i d'aquí en endavant tots els canvis en aquests directoris es guardaran en aquest volum. Tenir en compte que les rutes que apareixen en el fitxer `persistence.conf` no poden contenir espais en blanc o els components especials `.` i `...`. A més, ni `/lib`, `/lib/live` (o qualsevol dels seus subdirectoris) ni `/` es poden fer persistents utilitzant muntatges personalitzats. Com a solució per a aquesta limitació es pot afegir `/ union` al fitxer `persistence.conf` per a aconseguir una persistència completa.

### 10.3.2 Utilitzar diversos medis persistents

Hi ha diferents mètodes per utilitzar múltiples volums de persistència per a diferents casos d'ús. Per exemple, utilitzar diversos volums al mateix temps o seleccionar-ne només un, entre varis, per a fins molt específics.

585 Es poden utilitzar diversos volums diferents de muntatges personalitzats (amb els seus propis fitxers `persistence.conf` però si diversos volums fan que el mateix directori sigui persistent, només s'utilitzarà un d'ells. Si qualsevol dels dos muntatges són "imbricats" (és a dir, un és un sub-directori de l'altre) el directori pare es muntarà abans que el directori fill per a evitar que amb el muntatge un directori no sigui ocultat per l'altre. Els muntatges personalitzats imbricats són problemàtics si estan enumerats en el mateix fitxer `persistence.conf`. Veure la pàgina del manual `persistence.conf(5)` per a saber com manejar aquest cas, si realment es necessita (una pista: en general no cal fer-ho).

586 Un possible cas d'ús: Per a guardar les dades de l'usuari, és a dir `/home` i les dades del superusuari, és a dir `/root` en diferents particions, crear dues particions amb l'etiqueta `persistence` i afegir un fitxer `persistence.conf` en cadascuna d'aquesta manera `# echo "/home" > persistence.conf` per a la primera partició que guardarà els fitxers de l'usuari i `# echo "/root" > persistence.conf` per a la segona partició que emmagatzemarà els fitxers del superusuari. Finalment, utilitzar el paràmetre d'arrencada `persistence`.

587 Si un usuari necessita múltiples volums de persistència del mateix tipus per a diferents ubicacions o proves, com `private` i `work`, el paràmetre d'arrencada `persistence-label` utilitzat juntament amb el paràmetre d'arrencada `persistence` permetrà tenir diversos dispositius amb la mateixa, però única, persistència. Un exemple seria si un usuari vol utilitzar una partició amb persistència amb l'etiqueta `private` per a dades personals com els marcadors d'un navegador, utilitzaria els paràmetres d'arrencada: `persistence persistence-label=private`. I per emmagatzemar dades relacionades amb el treball, com a documents, projectes de recerca o d'un altre tipus, utilitzaria els paràmetres d'arrencada: `persistence persistence-label=work`.

588 És important recordar que aquests volums, `private` i `work`, també necessiten tenir un fitxer `persistence.conf` en la seva arrel. La pàgina de manual de *live-boot* conté més informació sobre com utilitzar aquestes etiquetes amb els noms més antics.

## 10.4 Persistència amb xifratge

589 Utilitzar la persistència vol dir que algunes dades sensibles poden quedar exposades a risc. Especialment si les dades persistents s'emmagatzemen en un dispositiu portàtil com una memòria USB o un disc dur extern. És llavors quan el xifrat entra en joc. Fins i tot si el procediment pot semblar complicat a causa de la quantitat de passos que s'han de fer, és molt fàcil manejar particions xifrades amb *live-boot*. Per a utilitzar **luks**, que és el tipus de xifrat compatible, es necessita instal·lar *cryptsetup* tant en la màquina que crearà la partició xifrada com en el sistema en viu amb que es va a utilitzar la partició persistent xifrada.

Per a instal·lar *cryptsetup* a la nostra màquina:

```
# apt-get install cryptsetup
```

Per a instal·lar *cryptsetup* en el sistema viu, afegir-lo a una `package-lists`:

```
$ lb config
$ echo "cryptsetup" > config/package-lists/encryption.list.chroot
```

Una cop tinguem el nostre sistema en viu amb *cryptsetup*, bàsicament, només hem de crear una nova partició, xifrar-la

i arrencar amb els paràmetres `persistence` i `persistence-encryption=luks`. Podríem haver anticipat aquest pas i afegit els paràmetres d'arrencada seguint el procediment habitual:

```
$ lb config --bootappend-live "boot=live components persistence persistence-encryption=luks"
```

Anem a entrar en els detalls per a tothom que no està familiaritzat amb el xifrat. En el següent exemple utilitzarem una partició en un dispositiu USB que correspon a `/dev/sdc2`. Tenir en compte que cal determinar quina partició és la que es va a utilitzar en cada cas específic.

El primer pas és connectar la memòria usb i determinar de quin dispositiu es tracta. La manera més recomanable per a llistar dispositius és utilitzar `ls -l /dev/disk/by-id`. Després d'això, crear una nova partició i, a continuació, xifrar-la amb una frase de contrasenya de la següent manera:

```
# cryptsetup --verify-passphrase luksFormat /dev/sdc2
```

A continuació, obrir la partició luks al mapeador de dispositius virtuals. Es pot utilitzar qualsevol nom que es desitgi. Aquí utilitzem **live** com a exemple:

```
# cryptsetup luksOpen /dev/sdc2 live
```

El següent pas és omplir el dispositiu amb zeros abans de crear el sistema de fitxers:

```
# dd if=/dev/zero of=/dev/mapper/live
```

Ara, estem preparats per a crear el sistema de fitxers. Noteu que estem afegint l'etiqueta `persistence` perquè el dispositiu es munti com a magatzem de persistència durant l'arrencada.

```
# mkfs.ext4 -L persistence /dev/mapper/live
```

Per continuar amb la nostra configuració, necessitem muntar el dispositiu, per exemple, a `/mnt`.

```
# mount /dev/mapper/live /mnt
```

I crear el fitxer `persistence.conf` a l'arrel de la partició. Això és, com s'ha explicat abans, estrictament necessari. Veure **«El fitxer persistence.conf»**.

```
# echo "/" union" > /mnt/persistence.conf
```

Desmuntar el punt de muntatge:

```
# umount /mnt
```

I opcionalment, encara que podria ser una bona manera de protegir les dades que acabem d'agregar a la partició, podem tancar el dispositiu:

```
# cryptsetup luksClose live
```

614

Anem a resumir el procés. Fins ara, hem creat un sistema viu capaç de manejar xifratge, que es pot copiar a una memòria USB com s'explica a [«Copiar una imatge ISO híbrida en un dispositiu USB»](#). També hem creat una partició xifrada, que es pot situar en la mateixa memòria USB per portar a tot arreu i hem configurat la partició xifrada per ser utilitzada com a magatzem de persistència. Així que ara, només hem de arrencar el sistema en viu. En el moment d'arrencar, *live-boot* ens preguntarà la frase de contrasenya i muntarà la partició xifrada per a ser utilitzada per a la persistència.

## Personalització de la imatge binària

## 11. Personalització de la imatge binària

### 11.1 Carregadors d'arrencada

*live-build* utilitza *syslinux* i alguns dels seus derivats (depenent del tipus d'imatge) com carregadors d'arrencada per defecte. Es poden personalitzar fàcilment per satisfer totes les necessitats.

Per a utilitzar un tema complet, copiar `/usr/share/live/build/-bootloaders` a `config/bootloaders` i editar els fitxers allí. Si no es vol modificar totes les configuracions dels carregadors d'arrencada disponibles, només cal utilitzar una còpia local personalitzada d'un dels carregadors, per exemple, copiar la configuració d'**isolinux** a `config/bootloaders/isolinux` ja és suficient, depenent del cas d'ús.

Quan es modifica un dels temes per defecte, si es vol utilitzar una imatge de fons personalitzada que es mostrarà juntament amb el menú d'arrencada, es pot afegir una imatge de 640x480 píxels. Aleshores, esborrar el fitxer `splash.svg`.

Hi ha moltes possibilitats a l'hora de fer canvis. Per exemple, els derivats de *syslinux* estan configurats per defecte amb un temps d'espera de 0 (zero) el que significa que faran una pausa indefinida en la seva pantalla inicial fins que es premi una tecla.

Per a modificar el temps d'espera d'arrencada d'una imatge `iso-hybrid` es pot editar el fitxer **isolinux.cfg** especificant el temps d'espera en unitats de segons 1/10. Un fitxer **isolinux.cfg** modificat per a arrencar després de cinc segons seria semblant a aquest:

```
include menu.cfg
default vesamenu.c32
prompt 0
timeout 50
```

### 11.2 metadades ISO

Quan es crea una imatge binària ISO9660, es poden utilitzar les següents opcions per a afegir diverses metadades textuais. Això pot ajudar a identificar fàcilment la versió o la configuració d'una imatge sense arrencar-la.

- **LB\_ISO\_APPLICATION/--iso-application NAME:** Ha de descriure l'aplicació que serà a la imatge. La longitud màxima per a aquest camp és de 128 caràcters.
- **LB\_ISO\_PREPARER/--iso-preparer NAME:** Ha de descriure al preparador de la imatge, en general amb algunes dades de contacte. El valor per defecte per a aquesta opció és la versió de *live-build* utilitzada, la qual cosa pot ajudar amb la depuració d'errors posterior. La longitud màxima per a aquest camp és de 128 caràcters.
- **LB\_ISO\_PUBLISHER/--iso-publisher NAME:** Ha de descriure l'editor de la imatge, en general amb algunes dades de contacte. La longitud màxima per a aquest camp és de 128 caràcters.
- **LB\_ISO\_VOLUME/--iso-volume NAME:** Això ha d'especificar l'ID de volum de la imatge. Això s'utilitza com una etiqueta visible per a l'usuari en algunes plataformes com Windows i Apple Mac OS. La longitud màxima per a aquest camp és de 32 caràcters.

## Personalització de l'instal·lador de debian

### 12. Personalització de l'instal·lador de debian

Les imatges del sistema en viu es poden integrar amb l'instal·lador de Debian. Hi ha un nombre de diferents tipus d'instal·lació, que varien en el que s'inclou i en com opera l'instal·lador.

Tenir en compte l'ús acurat de les lletres majúscules quan es refereix a "l'instal·lador de Debian" en aquesta secció - quan s'utilitza així ens referim explícitament a l'instal·lador normal del sistema Debian, i no a una altra cosa. Es veu sovint abreujat com "d-i".

#### 12.1 Tipus d'instal·lador de Debian

Els tres principals tipus d'instal·lador són els següents:

**Instal·lador de Debian "Normal"** : Aquesta és una imatge normal de sistema en viu amb un nucli i initrd independents que (quan es seleccionen des del carregador d'arrencada adequat) realitzen una instal·lació estàndard de Debian, igual que si s'hagués descarregat i arrencat una imatge de Debian des d'un CD. Les imatges que contenen un sistema viu i aquest tipus d'instal·lador independent s'anomenen sovint "imatges combinades".

Amb aquest tipus d'imatges, Debian s'instal·la descarregant i instal·lant paquets .deb mitjançant *debootstrap*, des dels medis locals o alguna xarxa, això resulta en un sistema Debian per defecte instal·lat al disc dur.

Tot aquest procés pot ser preconfigurat i personalitzat de moltes formes, veure les pàgines corresponents al manual de l'instal·lador de Debian per a més informació. Quan es té un fitxer de preconfiguració que funcioni, *live-build* pot posar-lo automàticament a la imatge i activar-lo.

**Instal·lador de Debian "Live"** : Aquesta és una imatge en viu amb un nucli i initrd independents que (quan es seleccionen des del carregador d'arrencada adequat) llancen un instal·lador de Debian.

La instal·lació continuarà de forma idèntica a l'instal·lació que s'ha descrit anteriorment, però en la fase d'instal·lació dels paquets, en lloc d'utilitzar *debootstrap* per a buscar-los i instal·lar-los, es copia el sistema de fitxers viu a la destinació. Això s'aconsegueix amb un udeb especial anomenat *live-installer*.

Després d'aquesta etapa, l'instal·lador de Debian continua de forma normal, instal·lant i configurant elements com ara els gestors d'arrencada i els usuaris locals, etc

**Nota:** per a donar suport a les entrades de l'instal·lador normal i live en el gestor d'arrencada del mateix medi s'ha de desactivar el *live-installer* mitjançant la preconfiguració *live-installer/enable=false*.

**Instal·lador de Debian "d'escriptori"** : Independentment del tipus d'instal·lador de Debian inclòs, es pot iniciar el d-i des de l'escriptori fent clic damunt una icona. Això és senzill per a l'usuari però perquè funcioni s'ha d'afegir el paquet *debian-installer-launcher*.

Tenir en compte que, per defecte, *live-build* no inclou imatges de l'instal·lador de Debian en les imatges, ha de ser específicament activat amb *lb config*. A més, tenir en compte que per a que funcioni l'instal·lador "d'escriptori" el nucli del sistema viu ha de coincidir amb el nucli que el d-i utilitza per a l'arquitectura especificada. Per exemple:

```
$ lb config --architectures i386 --linux-flavours 486 \
--debian-installer live
```



```
$ echo debian-installer-launcher >> config/package-lists/my.list.chroot
```

## 12.2 Personalització de l'instal·lador de Debian amb preconfiguració

Com es descriu en el Manual de l'instal·lador de Debian, a l'apèndix B a <https://www.debian.org/releases/stable/i386/apb.html>, “la preconfiguració proporciona una manera de respondre a les preguntes durant la instal·lació, sense haver d'introduir les respostes manualment mentre la instal·lació està en curs. Això permet automatitzar completament la majoria dels tipus d'instal·lacions i fins i tot ofereix algunes característiques no disponibles durant les instal·lacions normals.” Aquest tipus de personalització s'aconsegueix millor amb *live-build* col·locant la configuració en un fitxer `preseed.cfg` a `config/includes.installer/`. Per exemple, per a preconfigurar la variant local `en_US`:

```
$ echo "d-i debian-installer/locale string en_US" \  
>> config/includes.installer/preseed.cfg
```

## 12.3 Personalitzar el contingut de l'instal·lador de Debian

Per motius experimentals o de depuració d'errors, és possible que es vulgui incloure paquets `udeb` creats localment per al `d-i`. Per a afegir-los a la imatge posar-los a `config/packages.binary/`. Es poden incloure fitxers addicionals o de substitució i alguns directoris a l'`initrd` de l'instal·lador d'una manera similar a `<Live/chroot local includes>`, posant el material a `config/includes.installer/`.

## Projecte

## Contribuir al projecte

### 13. Contribuir al projecte

Quan s'envia una contribució, s'ha d'identificar clarament el titular dels drets d'autor i incloure la declaració de concessió de llicències aplicables. Recordar que per a ser acceptada, la contribució ha de tenir una llicència igual que la resta del document, a saber, la versió de la GPL 3 o superior.

Les contribucions al projecte, com ara traduccions i pegats, són molt benvingudes. Qualsevol persona pot fer un lliurament directe al repositori. No obstant això, demanem que s'enviïn els canvis grans a la llista de correu per a parlar-ne en primer lloc. Veure la secció [«Contacte»](#) per a més informació.

El Live Systems Project utilitza Git com a sistema de control de versions i gestió de codi font. Com s'explica en [«Repositoris Git»](#) hi ha dues branques principals de desenvolupament: **debian** i **debian-next**. Tothom pot fer lliuraments a les branques *debian-next* dels repositoris *live-boot*, *live-build*, *live-config*, *live-images*, *live-manual* i *live-tools*.

No obstant això, hi ha certes restriccions. El servidor rebutja:

- Push que no són fast-forward.
- Commits merge.
- Afegir o eliminar etiquetes o branques.

Tot i que tots els lliuraments poden ser revisats, demanem que s'utilitzi el sentit comú i es facin bons lliuraments amb bons missatges.

- Escriure missatges de lliurament que consisteixen en oracions completes i significatives en anglès, començant amb una lletra majúscula i acabant amb un punt. En general, aquests

començaran amb la forma 'Fixing/Adding/Removing/Correcting/-Translating/...'.

- Escriure bons missatges de lliurament. La primera línia ha de ser un resum exacte dels continguts del lliurament, que s'inclourà en la llista de canvis. Si es necessita fer algunes explicacions més, escriure a sota deixant una línia en blanc després de la primera línia i després una altra línia en blanc després de cada paràgraf. Les línies dels paràgrafs no han de superar els 80 caràcters de longitud.
- Fer lliuraments de manera atòmica, és a dir, no barrejar coses no relacionades en el mateix lliurament. Fer un lliurament diferent per a cada canvi que es faci.

#### 13.1 Fer canvis

Per tal de fer un push als repositoris, s'ha de seguir el següent procediment. Aquí s'utilitza *live-manual* com a exemple, per tant, cal substituir-lo pel nom del repositori amb que es desitja treballar. Per a obtenir informació detallada sobre com editar *live-manual* veure [«Contribuir a aquest document»](#).

- Obtenir la clau pública:

```
$ mkdir -p ~/.ssh/keys
$ wget http://live-systems.org/other/keys/git@live-systems.org -O ~/.ssh/keys/git@live-systems.org
$ wget http://live-systems.org/other/keys/git@live-systems.org.pub -O ~/.ssh/keys/git@live-systems.org.pub
$ chmod 0600 ~/.ssh/keys/git@live-systems.org*
```

- Afegir la següent secció a la configuració del vostre openssh-client:

```
$ cat >> ~/.ssh/config << EOF
Host live-systems.org
  Hostname live-systems.org
  User git
  IdentitiesOnly yes
  IdentityFile ~/.ssh/keys/git@live-systems.org
EOF
```

```
$ git push
```

- Fer una còpia del manual a través de ssh:

```
$ git clone git@live-systems.org:/live-manual.git
$ cd live-manual && git checkout debian-next
```

- Assegurar-se de tenir el autor i el correu electrònic configurats al Git:

```
$ git config user.name "John Doe"
$ git config user.email john@example.org
```

**Important:** Tenir en compte que s'han d'enviar els canvis a la branca **debian-next** .

- Fer els canvis. En aquest exemple s'hauria d'escriure primer una nova secció sobre aplicar pegats i després preparar-se per a afegir els fitxers i escriure el missatge de la següent manera:

```
$ git commit -a -m "Adding a section on applying patches."
```

- Fer un push al servidor:

## 680 Informar dels errors

### 681 14. Informar dels errors

682 Live systems està lluny de ser perfecte, però volem que sigui el més perfecte possible - amb la vostra ajuda. No dubtar d'informar sobre un error. És millor omplir un informe dues vegades que mai. No obstant això, aquest capítol inclou recomanacions sobre com presentar bons informes d'errors.

683 Per als impacients

- 684 • Sempre consultar primer les actualitzacions del estat de la imatge a la nostra pàgina web a <http://live-systems.org/> per a veure els problemes coneguts.
- 685 • Abans d'enviar un informe d'errors, sempre tractar de reproduir l'error amb **les versions més recents** de la branca de *live-build*, *live-boot*, *live-config* i *live-tools* què utilitzeu (com la darrera versió 4.x de *live-build* si s'utilitza *live-build* 4).
- 686 • Intentar donar **la informació més específica possible** sobre l'error. Això inclou (almenys) la versió de *live-build*, *live-boot*, *live-config* i *live-tools* i la distribució del sistema en viu que s'està construint.

#### 687 14.1 Problemes coneguts

688 Ja que les distribucions Debian **testing** i Debian **unstable** són blancs mòbils, quan s'especifica una d'elles com sistema de destinació, no sempre és possible construir amb èxit.

689 Si això és massa difícil, no construir un sistema basat en **testing** o **unstable**, sinó més aviat, utilitzar **stable**. *live-build* sempre construeix la versió **stable** per defecte.

690 El problemes coneguts es mostren sota la secció 'status' a la nostra

pàgina web a <http://live-systems.org/>.

Està fora de l'abast d'aquest manual ensenyar a identificar correctament i solucionar els problemes dels paquets de les distribucions en desenvolupament, però, hi ha dues coses que sempre es pot provar: Si la construcció falla quan la distribució de destinació és **testing**, provar **unstable**. Si **unstable** tampoc funciona, tornar a **testing** i fer un pin de la versió més recent del paquet que falla de **unstable** (veure **<APT pinning>** per a més detalls).

### 692 14.2 Reconstruir des de zero

Per a assegurar-se que un error en particular no és causat per un sistema mal construït, reconstruir sempre tot el sistema en viu a partir de zero per veure si l'error és reproducible.

### 694 14.3 Fer servir paquets actualitzats

La utilització de paquets obsolets pot causar problemes significatius al tractar de reproduir (i en última instància, arreglar) el problema. Comprovar que el sistema de construcció està actualitzat i tots els paquets inclosos en la imatge estan també actualitzats.

### 696 14.4 Recopilar informació

Proporcionar informació suficient amb l'informe. Incloure, com a mínim, la versió exacta de *live-build* i els passos per a reproduir-lo. Utilitzar el sentit comú i proporcionar tota altra informació pertinent si es pensa que això pot ajudar a resoldre el problema.

Per a treure el màxim profit del informe d'errors, es requereix com a mínim la informació següent:

- Arquitectura del sistema amfitrió
- Distribució del sistema amfitrió
- Versió de *live-build* al sistema amfitrió
- Versió de Python al sistema amfitrió
- Versió de *debootstrap* i/o *cdebootstrap* al sistema amfitrió
- Arquitectura del sistema en viu
- Distribució del sistema en viu
- Versió de *live-boot* al sistema amfitrió
- Versió de *live-config* al sistema amfitrió
- Versió de *live-tools* al sistema amfitrió

Es pot generar un log del procés de construcció mitjançant l'ordre `tee`. Recomanem fer-ho automàticament amb un script `auto/build` (veure [«Gestió d'una configuració»](#) per a més detalls).

```
# lb build 2>&1 | tee build.log
```

Durant l'arrencada, *live-boot* i *live-config* emmagatzemen els seus logs a `/var/log/live/`. Comprovar aquest fitxers per a detectar missatges d'error.

A més, per a descartar altres errors, sempre és una bona idea comprimir el directori `config/` i pujar-lo a algun lloc (**no** enviar-lo com arxiu adjunt a la llista de correu), perquè puguem tractar de reproduir els errors que s'han trobat. Si això és difícil (per exemple, a causa de la mida del arxiu) es pot utilitzar la sortida de `lb config --dump` que produeix un resum del arbre de configuració (és a dir, fa un llistat dels fitxers dins els subdirectoris de `config/`, però no els inclou).

Recordar que s'ha d'enviar qualsevol log que es produeixi amb la configuració regional en anglès, per exemple, executar les ordres de *live-build* començant per `LC_ALL=C` o `LC_ALL=en_US`.

## 14.5 Aïllar el cas que falla, si és possible

Si pot ser, aïllar el cas que falla al canvi més petit possible que fa que no funcioni. No sempre és fàcil fer això, per tant, si no es possible fer-ho pel informe, no preocupar-se. No obstant això, si es planeja bé el cicle de desenvolupament, i s'utilitzen petits conjunts de canvis suficients per iteració, es pot ser capaç d'aïllar el problema mitjançant la construcció d'una configuració 'base' més senzilla que s'ajusti a la configuració desitjada més el conjunt de canvis que fa que no funcioni. Si es difícil classificar quins canvis fan que falli, pot ser que s'inclogui massa en cada conjunt de canvis i s'ha de desenvolupar en petits increments.

## 14.6 Utilitzar el paquet correcte per a informar de l'error

Si no és clar quin component és el responsable de l'error o si l'error és general pel que fa als sistemes vius, es pot omplir un informe d'errors sobre el pseudopaquet `debian-live`.

No obstant això, estarem molt agraïts si s'intenta limitar la recerca segons el lloc on apareix l'error.

### 14.6.1 A l'hora de construir mentre bootstrapping

*live-build* crea primer un sistema Debian bàsic amb *debootstrap* o *cdebootstrap*. Depenent de l'eina utilitzada i la distribució Debian que s'està creant mitjançant el bootstrapping, pot fallar. Si un error apareix aquí, comprovar si l'error està relacionat amb un paquet

específic de Debian (el més probable), o si està relacionat a l'eina bootstrapping en si mateixa.

721 En ambdós casos, això no és un error del sistema en viu, sinó de Debian en si mateix i, probablement, no ho podem arreglar directament. Informar del error sobre l'eina de debootstrapping o el paquet que falla.

#### 722 14.6.2 A l'hora de construir, durant la instal·lació de paquets

723 *live-build* instal·la paquets addicionals del arxiu de Debian i en funció de la distribució Debian utilitzada i de l'estat diari de l'arxiu, pot fallar. Si un error apareix aquí, comprovar si l'error és també reproducible en un sistema normal.

724 Si aquest és el cas, no es tracta d'un error del sistema en viu, sinó de Debian - Informar d'això sobre el paquet que falla. Executar *debootstrap* per separat de la construcció del sistema Live o executar `lb bootstrap --debug` per a tenir més informació.

725 A més, si es fa servir un mirall local i/o qualsevol tipus de proxy i s'està experimentant algun problema, sempre s'ha de mirar de reproduir-lo fent un bootstrapping a partir d'un mirall oficial.

#### 726 14.6.3 En el moment d'arrencar

727 Si la imatge no arrenca, informar a la llista de correu, juntament amb la informació sol·licitada a «[Recopilar informació](#)». No oblidar-se d'esmentar, com/quan la imatge falla, ja sigui amb virtualització o maquinari real. Si s'utilitza una tecnologia de virtualització d'algun tipus, sempre fer la prova amb maquinari real abans d'informar d'un error. Proporcionar una captura de pantalla de l'error és també molt útil.

#### 14.6.4 En temps d'execució

728

Si un paquet s'ha instal·lat correctament, però falla quan s'executa el sistema Live, això és probablement un error al sistema en viu. No obstant això:

729

#### 14.7 Fer la recerca

730

Abans de presentar l'informe d'errors, cercar a la web el missatge d'error o símptoma que s'està rebent. Ja que és molt poc probable que sigui l'única persona que té un problema en particular. Sempre hi ha una possibilitat que hagi estat discutit en un altre lloc i hi hagi una possible solució, pegat o s'hagi proposat una solució alternativa.

731

S'ha de prestar especial atenció a la llista de correu dels sistemes en viu, així com a la pàgina web, ja que és probable que continguin la informació més actualitzada. Si aquesta informació existeix, incloure una referència a aquesta en l'informe d'errors.

732

A més, s'hauria de comprovar les llistes d'errors actuals de *live-build*, *live-boot*, *live-config* i *live-tools* per a veure si ja s'ha informat sobre alguna cosa semblant .

733

#### 14.8 On informar dels errors

734

El Live Systems Project manté un registre de tots els errors en el sistema de seguiment d'errors de Debian (BTS). per a obtenir informació sobre la utilització del sistema, es pot consultar <<https://bugs.debian.org/>>. També es poden enviar els informes dels errors mitjançant l'ordre `reportbug` del paquet amb el mateix nom.

735

En general, s'ha d'informar dels errors en temps de construcció contra el paquet *live-build*, dels errors durant l'arrencada contra

736

*live-boot* i dels errors de temps d'execució contra *live-config*. Si no s'està segur de quin paquet és l'adequat o es necessita més ajuda abans d'enviar un informe d'errors, informar contra el pseudopaquet *debian-live*. Nosaltres ens farem càrrec d'ell i el reassignarem on sigui procedent.

737

Tenir en compte que els errors trobats en les distribucions derivades de Debian (com Ubuntu i altres) no han de ser enviats al BTS de Debian tret que puguin ser reproduïts també en sistemes Debian utilitzant paquets oficials de Debian.



## Estil de Codi

### 15. Estil de Codi

En aquest capítol es documenta l'estil de codi utilitzat a live systems.

#### 15.1 Compatibilitat

- No utilitzar una sintaxi o semàntica que sigui exclusiva de l'interpret d'ordres Bash. Per exemple, l'ús dels arrays.
- Utilitzar només el subconjunt POSIX - per exemple, utilitzar `$(foo)` en lloc de `'foo'`.
- Es pot comprovar els scripts amb `'sh -n'` i `'checkbashisms'`.
- Assegurar-se que tot el codi funciona amb `'set -e'`.

#### 15.2 Indentació

- Utilitzar sempre tabuladors en lloc d'espais.

#### 15.3 Ajust de línia

- En general, les línies són de 80 caràcters com a màxim.
- Utilitzar "l'estil Linux" de salts de línia:

Mal:

```
if foo; then
    bar
fi
```

Bé:

```
if foo
then
    bar
fi
```

- El mateix val per a les funcions:

Mal:

```
Foo () {
    bar
}
```

Bé:

```
Foo ()
{
    bar
}
```

#### 15.4 Variables

- Les variables van sempre en majúscules.
- Les variables que s'utilitzen a *live-build* sempre comencen amb el prefix `LB_`
- Les variables temporals internes de *live-build* comencen amb el prefix `<=underscore>LB_`

- Les variables locals comencen amb el prefix *live-build* 764  
`<=underscore><=underscore>LB_`

- Les variables en relació a un paràmetre d'arrencada de *live-config* comencen amb `LIVE_`. 765

- Totes les altres variables de *live-config* comencen amb el prefix 766  
`_`

- Utilitzar claus al voltant de les variables, per exemple, escriure 767  
`${Foo}` en lloc de `$Foo`.

- Protegir sempre les variables amb cometes per a respectar els 768  
 espais en blanc potencials: escriure `"${Foo}"` no `${Foo}`.

- Per raons de coherència, utilitzar sempre cometes al assignar 769  
 valors a les variables:

Mal:

```
F00=bar
```

Bé:

```
F00="bar"
```

- Si s'utilitzen múltiples variables, posar cometes a l'expressió 774  
 completa:

Mal:

```
if [ -f "${F00}"/foo/"${BAR}"/bar ]
then
    foobar
```

```
fi
```

Bé:

```
if [ -f "${F00}/foo/${BAR}/bar" ]
then
    foobar
fi
```

## 15.5 Miscel·lània

- Utilitzar `|` (sense les cometes) com separador en l'ús de `sed`, 780  
 per exemple, `"sed -e `s|'|`"` (sense `""`).
- No utilitzar l'ordre `test` per a fer comparacions o tests, utilitzar 781  
`"[" "` (sense `""`); per exemple, `"if [ -x /bin/foo ]; ..."` i no  
`"if test -x /bin/foo; ..."`.
- Utilitzar `case` sempre que sigui possible en lloc de `test`, ja que 782  
 és més fàcil de llegir i més ràpid en l'execució.
- Fer servir noms en majúscula per a les funcions per evitar 783  
 conflictes amb l'entorn dels usuaris.

## Procediments

### 16. Procediments

Aquest capítol documenta els procediments dins del Live Systems Project per a les diferents tasques que necessiten la cooperació amb altres equips de Debian.

#### 16.1 Publicacions majors

El llançament d'una nova versió de Debian inclou una gran quantitat de diferents equips que treballen junts per a fer que això succeeixi. En algun moment, l'equip Live arriba i construeix imatges en viu del sistema. Els requisits per a fer això són:

- Un mirall que contingui les versions publicades dels arxius de debian i debian-security on pugui accedir el build de debian-live.
- S'ha de conèixer el nom de la imatge (per exemple, debian-live-VERSION-ARCH-FLAVOUR.iso).
- S'han de sincronitzar les dades de debian-cd (udeb exclude lists).
- Les imatges es construeixen i s'en fa una rèplica a cdimage.debian.org.

#### 16.2 Publicacions puntuals

- Un cop més, necessitem miralls actualitzats de debian i debian-security.
- Les imatges es construeixen i s'en fa una rèplica a cdimage.debian.org.
- Enviar un anunci per correu electrònic.

#### 16.2.1 Última publicació puntual d'una versió de Debian

Recordar que s'han d'ajustar els miralls chroot i binary en la construcció de l'última sèrie d'imatges per a una versió de Debian després de canviar-les de ftp.debian.org a archive.debian.org. D'aquesta manera, les imatges prefabricades ja velles continuaran sent útils sense modificacions per part dels usuaris.

#### 16.2.2 Plantilla per a anunciar una publicació puntual

Es pot generar un correu per a anunciar una publicació puntual mitjançant la plantilla següent i l'ordre:

```
$ sed \
-e 's|@MAJOR@|7.0|g' \
-e 's|@MINOR@|7.0.1|g' \
-e 's|@CODENAME@|wheezy|g' \
-e 's|@ANNOUNCE@|2013/msgXXXXX.html|g'
```

Llegir el correu acuradament abans d'enviar-lo i passar-lo als altres per a la correcció d'errades.

```
Updated Live @MAJOR@: @MINOR@ released

The Live Systems Project is pleased to announce the @MINOR@ update of the
Live images for the stable distribution Debian @MAJOR@ (codename "  
@CODENAME@").

The images are available for download at:

<http://live-systems.org/cdimage/release/current/>

and later at:

<http://cdimage.debian.org/cdimage/release/current-live/>
```

This update includes the changes of the Debian @MINOR@ release:

<<https://lists.debian.org/debian-announce/@ANNOUNCE@>>

Additionally it includes the following Live-specific changes:

- \* [INSERT LIVE-SPECIFIC CHANGE HERE]
- \* [INSERT LIVE-SPECIFIC CHANGE HERE]
- \* [LARGER ISSUES MAY DESERVE THEIR OWN SECTION]

### About Live Systems

-----

The Live Systems Project produces the tools used to build official live systems and the official live images themselves for Debian.

### About Debian

-----

The Debian Project is an association of Free Software developers who volunteer their time and effort in order to produce the completely free operating system Debian.

### Contact Information

-----

For further information, please visit the Live Systems web pages at <<http://live-systems.org/>>, or contact the Live Systems team at <[debian-live@lists.debian.org](mailto:debian-live@lists.debian.org)>.

## Repositoris Git

### 17. Repositoris Git

La llista de tots els repositoris del Live Systems Project és a <http://live-systems.org/gitweb/>. Les URLs git del projecte tenen la forma: `protocol://live-systems.org/git/repositori`. Per tant, per a clonar *live-manual* en només lectura, llançar:

```
$ git clone git://live-systems.org/git/live-manual.git
```

O,

```
$ git clone https://live-systems.org/git/live-manual.git
```

O,

```
$ git clone http://live-systems.org/git/live-manual.git
```

Les adreces per a clonar amb permís d'escriptura tenen la forma: `git@live-systems.org/repositori`.

Així que, de nou, per a clonar *live-manual* a través de ssh escriure:

```
$ git clone git@live-systems.org:live-manual.git
```

L'arbre git es compon de diverses branques diferents. Les

branques **debian** i **debian-next** són particularment notables perquè contenen el treball real que, amb el temps, serà inclòs en cada nova versió.

Després de clonar qualsevol dels repositoris existents, estarem a la branca **debian**. Això és apropiat per a fer una ullada a l'estat de l'última versió del projecte, però abans de començar a treballar és fonamental canviar a la branca **debian-next**. Per fer això:

```
$ git checkout debian-next
```

La branca **debian-next**, que no sempre és fast-forward, és on es realitzen tots els canvis abans de fusionar-los amb la branca **debian**. Per fer una analogia, és com un camp de proves. Si s'està treballant en aquesta branca i es necessari fer un pull, s'haurà de fer un `git pull --rebase` perquè les modificacions locals es guardin mentre s'actualitza des del servidor i llavors els canvis locals es posaran al damunt de tot.

#### 17.1 Gestió de múltiples repositoris

Si es té la intenció de clonar diversos repositoris de Live Systems i canviar a la branca **debian-next** immediatament per a comprovar l'últim codi, escriure un pegat o contribuir amb una traducció, s'ha de saber que el servidor git proporciona un fitxer `mrconfig` per a facilitar el maneig de múltiples repositoris. Per a utilitzar-lo cal instal·lar el paquet *mr* i després d'això, fer:

```
$ mr bootstrap http://live-systems.org/other/mr/mrconfig
```

Aquesta ordre automàticament clonarà i canviarà a la branca

**debian-next** els repositoris de desenvolupament dels paquets debian produïts pel projecte. Aquests inclouen, entre d'altres, el repositori *live-images*, que conté les configuracions utilitzades per a construir les imatges prefabricades que el projecte publica per a l'ús general. per a obtenir més informació sobre com utilitzar aquest repositori, consultar [«Clonar una configuració publicada via Git»](#)

## Exemples

## Exemples

### 18. Exemples

En aquest capítol s'inclouen exemples de construccions per a casos d'ús específics amb sistemes en viu. Si s'és nou en la construcció d'imatges en viu pròpies, us suggerim mirar els tres tutorials en seqüència, ja que cada un ensenya noves tècniques que ajuden a utilitzar i entendre els exemples restants.

#### 18.1 Ús dels exemples

per a utilitzar aquests exemples es necessita un sistema de construcció que compleixi les exigències enumerades a **«Requisits»** y que tingui *live-build* instal·lat com es descriu a **«Instal·lació de live-build»**.

Cal notar que, per a abreujar, en aquests exemples no s'especifica un mirall local per a utilitzar en la construcció. Es poden accelerar les descàrregues considerablement si s'utilitza un mirall local. Es pot especificar les opcions quan s'utilitza `lb config`, com es descriu a **«Miralls de distribució utilitzats en temps de construcció»**, o per a major comoditat, establir el valor predeterminat per al sistema de construcció en el fitxer `/etc/live/build.conf`. Només cal crear aquest fitxer i establir a les variables al mirall preferit. Tots els altres miralls que s'utilitzin en la construcció adoptaran valors per defecte segons aquests valors. Per exemple:

```
LB_MIRROR_BOOTSTRAP="http://mirror/debian/"
LB_MIRROR_CHROOT_SECURITY="http://mirror/debian-security/"
LB_MIRROR_CHROOT_BACKPORTS="http://mirror/debian-updates/"
```

### 18.2 Tutorial 1: Una imatge per defecte

**Cas d'ús:** Crear una primera imatge senzilla, aprenent els conceptes bàsics de *live-build*.

En aquest tutorial, anem a construir una imatge ISO híbrida per defecte que contingui només els paquets de base (no té Xorg) i altres paquets de suport de sistema en viu, com un primer exercici en l'ús de *live-build*.

No pot ser més senzill que això:

```
$ mkdir tutorial1 ; cd tutorial1 ; lb config
```

Examinar el contingut del directori `config/` si es vol. Es veurà emmagatzemada aquí una configuració en esquelet, a punt per a ser personalitzada o, en aquest cas, per a ser utilitzada immediatament per a construir una imatge per defecte.

Ara, com a superusuari, construir la imatge, guardant un log del que es construeix amb `tee`.

```
# lb build 2>&1 | tee build.log
```

Suposant que tot va bé, després d'una estona, el directori actual contindrà una `live-image-i386.hybrid.iso`. Aquesta imatge ISO híbrida es pot arrencar en una màquina virtual tal com s'explica a **«Provar una imatge ISO amb Qemu»** i **«Provar una imatge ISO amb VirtualBox»**, o bé copiada a un dispositiu USB com es descriu a **«Gravar una imatge ISO en un medi físic»** i **«Copiar una imatge ISO híbrida en un dispositiu USB»**, respectivament.



## 18.3 Tutorial 2: Una utilitat de navegador web

840

**Cas d'ús:** Crear una imatge d'una utilitat de navegador web, aprenent a aplicar personalitzacions.

En aquest tutorial, anem a crear una imatge adequada per al seu ús com a una utilitat de navegador web, que serveix com introducció a la personalització de les imatges de sistemes en viu.

```
$ mkdir tutorial2
$ cd tutorial2
$ lb config
$ echo "task-lxde-desktop iceweasel" >> config/package-lists/my.list.chroot
$ lb config
```

La nostra elecció de LXDE per a aquest exemple reflecteix el nostre desig d'oferir un entorn d'escriptori mínim, ja que l'objectiu de la imatge és l'únic ús que tenim al cap, el navegador web. Podríem anar encara més lluny i oferir una configuració per defecte per al navegador web a config/includes.chroot/etc/iceweasel/profile/, o paquets addicionals de suport per a la visualització de diversos tipus de contingut web, però deixem això com a un exercici per al lector.

Construir la imatge, de nou com a superusuari, guardant un log com al **<Tutorial 1>**:

```
# lb build 2>&1 | tee build.log
```

Un cop més, verificar que la imatge està bé i provar-la, com al **<Tutorial 1>**.

## 18.4 Tutorial 3: Una imatge personalitzada

**Cas d'ús:** Crear un projecte per a construir una imatge personalitzada, que contingui el programari favorit per a portar en una memòria USB allà on es vagi i que evolucionarà en revisions successives tal i com les necessitats i les preferències canvien.

Com la nostra imatge personalitzada canviarà durant un nombre de revisions i volem fer un seguiment d'aquests canvis, provar coses experimentals i possiblement revertir-les si les coses no surten bé, anem a mantenir la nostra configuració en el popular sistema de control de versions git. També utilitzarem les millors pràctiques de configuració automàtica mitjançant scripts auto com s'explica a **<Gestió d'una configuració>**.

### 18.4.1 Primera revisió

```
$ mkdir -p tutorial3/auto
$ cp /usr/share/doc/live-build/examples/auto/* tutorial3/auto/
$ cd tutorial3
```

Editar auto/config de la manera següent:

```
#!/bin/sh

lb config noauto \
    --architectures i386 \
    --linux-flavours 686-pae \
    "${@}"
```

Executar `lb config` per a crear l'arbre de configuració, utilitzant el

script `auto/config` que s'acaba de crear:

```
$ lb config
```

Ara, omplir la llista local de paquets:

```
$ echo "task-lxde-desktop iceweasel xchat" >> config/package-lists/my.list.chroot
```

En primer lloc, amb `--architectures i386` s'assegura que al nostre sistema de construcció `amd64` podem construir una versió de 32 bits adequada per al seu ús en la majoria de màquines. En segon lloc, fem servir `--linux-flavours 686-pae` perquè no creiem que utilitzarem aquesta imatge en sistemes molt més vells. En tercer lloc, hem triat la tasca metapaquet `lxde` per a donar-nos un escriptori mínim. I, finalment, hem afegit dos paquets inicials favorits: *iceweasel* i *xchat*.

Ara, construir la imatge:

```
# lb build
```

Tenir en compte que a diferència dels dos primers tutorials, ja no s'ha d'escriure `2>&1 |tee build.log` ja que ara s'inclou al script `auto/build`.

Quan s'ha provat la imatge (com al [Tutorial 1](#)) i s'està satisfet del seu funcionament, és el moment d'iniciar el repositori `git`, afegint només els scripts `auto` que s'han creat, i llavors fer el primer lliurament:

```
$ git init
$ cp /usr/share/doc/live-build/examples/gitignore .gitignore
$ git add .
$ git commit -m "Initial import."
```

## 18.4.2 Segona revisió

En aquesta revisió, anem a netejar després de la primera construcció, afegir el paquet *vlc* a la nostra configuració, reconstruir, provar i fer el lliurament.

L'ordre `lb clean` netejarà tots els fitxers generats en la construcció anterior a excepció del cache, el que estalvia haver de tornar a descarregar els paquets. Això assegura que el `lb build` següent tornarà a executar totes les etapes per a regenerar els fitxers de la nostra nova configuració.

```
# lb clean
```

Ara afegim el paquet *vlc* al llistat de paquets local a `config/package-lists/my.list.chroot`:

```
$ echo vlc >> config/package-lists/my.list.chroot
```

Construir de nou:

```
# lb build
```

Provar, i quan s'estigui satisfet, fer el lliurament de la propera

revisió:

```
$ git commit -a -m "Adding vlc media player."
```

Per descomptat, es possible fer canvis més complicats en la configuració, potser afegint fitxers en els subdirectoris de `config/`. Quan es fa un lliurament de les noves revisions, s'ha de tenir cura de no editar a mà o incloure en el lliurament els fitxers de nivell superior de `config` que contenen variables `LB_*`, ja que són productes de construcció també, i sempre són netejats per `lb clean` i tornats a crear per `lb config` a través dels seus respectius scripts `auto`.

Hem arribat al final de la nostra sèrie de tutorials. Molts més tipus de personalització són possibles, amb les poques característiques explorades en aquests senzills exemples, es poden crear una varietat gairebé infinita d'imatges diferents. Els exemples que queden d'aquesta secció tracten diferents casos d'ús extrets de les experiències recollides dels usuaris de sistemes en viu.

## 18.5 Un client per a un quiosc VNC

**Cas d'ús:** Crear una imatge amb *live-build* per a connectar-se directament a un servidor VNC al arrencar.

Fer un directori de treball i crear una configuració d'esquelet en el seu interior, desactivant els «`recommends`» per a fer un sistema mínim. I a continuació, crear dues llistes inicials de paquets: La primera generada per un script proporcionat per *live-build* anomenat `Packages` (veure «[Generar llistes de paquets](#)»), i la segona incloent-hi *xorg*, *gdm3*, *metacity* i *xvnc4viewer*.

```
$ mkdir vnc-kiosk-client
$ cd vnc-kiosk-client
$ lb config -a i386 -k 686-pae --apt-recommends false
$ echo '! Packages Priority standard' > config/package-lists/standard.list.<
chroot
$ echo "xorg gdm3 metacity xvnc4viewer" > config/package-lists/my.list.<
chroot
```

Com s'explica a «[Afinar APT per a estalviar espai](#)» pot ser que s'hagi de tornar a afegir alguns paquets recomanats per a fer que la imatge funcioni correctament.

Una manera fàcil d'enumerar els `recommends` és utilitzar *apt-cache*. Per exemple:

```
$ apt-cache depends live-config live-boot
```

En aquest exemple, ens vam assabentar que havíem de tornar a incloure diversos paquets recomanats per *live-config* i *live-boot*: *user-setup* perquè funcioni l'autologin i *sudo* com a programa essencial per a apagar el sistema. A més, podria ser útil afegir *live-tools* per a poder copiar la imatge en la memòria RAM i *eject* per a expulsar, finalment, el medi en viu. Per tant:

```
$ echo "live-tools user-setup sudo eject" > config/package-lists/recommends<
.list.chroot
```

Després, crear el directori `/etc/skel` a `config/includes.chroot` i posar un fitxer `.xsession` personalitzat per a l'usuari per defecte que posarà en marxa *metacity* i iniciarà *xvncviewer*, connectant al port 5901 d'un servidor ubicat a 192.168.1.2:

```
$ mkdir -p config/includes.chroot/etc/skel
$ cat > config/includes.chroot/etc/skel/.xsession << EOF
#!/bin/sh

/usr/bin/metacity &
/usr/bin/xvncviewer 192.168.1.2:1

exit
EOF
```

Construir la imatge:

```
# lb build
```

Gaudir-ne.

## 18.6 Una imatge bàsica per a un dispositiu USB de 128MB

**Cas d'ús:** Crear una imatge per defecte amb alguns components eliminats per tal que càpiga en una clau USB de 128MB amb un petit espai de sobres per a utilitzar-lo com millor us sembli.

Al optimitzar una imatge per a adaptar-la a una mida determinada, cal comprendre la compensació que s'estan fent entre la mida i la funcionalitat. En aquest exemple, retallem tant només per a donar cabuda a material addicional dins d'una mida de 128MB, però sense fer res per a destruir la integritat dels paquets continguts, com la depuració de les dades de localització a través del paquet *localepurge* o altres optimitzacions "intrusives". De particular interès, utilitzem `--debootstrap-options` per a crear un sistema mínim des de zero.

```
$ lb config -k 486 --apt-indices false --apt-recommends false --debootstrap-  
-options "--variant=minbase" --firmware-chroot false --memtest none
```

Per a fer que la imatge funcioni correctament, hem de tornar a afegir, com a mínim, dos paquets recomanats, que es queden fora per l'opció `--apt-recommends false`. Veure [«Afinar APT per a estalviar espai»](#)

```
$ echo "user-setup sudo" > config/package-lists/recommends.list.chroot
```

Ara, crear la imatge de la forma habitual:

```
# lb build 2>&1 | tee build.log
```

En el sistema de l'autor en el moment d'escriure això, la configuració anterior produeix una imatge de 77MB. Això es compara favorablement amb la imatge de 177MB produïda per la configuració per defecte del [«Tutorial 1»](#).

El estalvi d'espai més gran aquí, en comparació amb la construcció d'una imatge per defecte en un sistema d'arquitectura i386, és seleccionar només la variant del nucli 486 en comptes del predeterminat `-k "486 686-pae"`. Deixar els índexs d'APT amb `--apt-indices false` també permet estalviar una bona quantitat d'espai, el desavantatge és que es necessita fer `apt-get update` abans d'utilitzar `apt` en el sistema en viu. Deixar els paquets recomanats amb `--apt-recommends false` estalvia una mica d'espai addicional, a costa d'ometre alguns paquets que d'una altra manera es podria esperar que hi fossin. `--debootstrap-options "--variant=minbase"` preinstal·la un sistema mínim des del principi. Al no incloure automàticament paquets de firmware

amb `--firmware-chroot false` també es guanya una mica d'espai. I finalment, `--memtest none` impedeix la instal·lació d'un comprovador de memòria.

**Nota:** Un sistema mínim es pot aconseguir també fent servir un script ganxo, com ara el `stripped.hook.chroot` de `/usr/share/doc/live-build/examples/hooks`. Es pot guanyar petites quantitats addicionals d'espai i produir una imatge de 62MB. No obstant, el script ganxo aconsegueix això eliminant documentació i altres fitxers dels paquets instal·lats al sistema. Això viola la integritat dels paquets i com és comenta a l'encapçalament del script, pot tenir conseqüències imprevistes. És per això que l'ús d'un *debootstrap* mínim és el mètode recomanat per a aconseguir aquest objectiu.

## 18.7 Un escriptori GNOME localitzat i amb instal·lador

**Cas d'ús:** Crear una imatge amb l'escriptori GNOME, localitzat per Suïssa i que inclogui un instal·lador.

Volem fer una imatge ISO híbrida per a l'arquitectura `i386` fent servir el nostre escriptori preferit, en aquest cas GNOME, que conté tots els mateixos paquets que serien instal·lats per l'instal·lador estàndard de Debian per a GNOME.

El nostre primer problema és descobrir els noms de les tasques del llenguatge apropiades. En l'actualitat, *live-build* no ens pot ajudar amb això. Tot i que es podria tenir sort i trobar-ho per assaig i error, hi ha una eina, `grep-dctrl`, per extreure les descripcions de les tasques de `tasksel-data`. Per a preparar-ho tot, assegurar-se de tenir totes dues coses:

```
# apt-get install dctrl-tools tasksel-data
```

Ara podem buscar les tasques apropiades, primer amb:

```
$ grep-dctrl -Ftest-lang de /usr/share/tasksel/descs/debian-tasks.desc -<↵
sTask
Task: german
```

Mitjançant aquesta ordre, es descobreix que la tasca s'anomena, amb suficient claredat, `german`. Ara, per a trobar les tasques relacionades:

```
$ grep-dctrl -Fenhances german /usr/share/tasksel/descs/debian-tasks.desc -<↵
sTask
Task: german-desktop
Task: german-kde-desktop
```

En el moment d'arrencar es generarà la variant regional **de\_CH.UTF-8** i es seleccionarà la disposició del teclat **ch**. Ara posarem les peces juntes. Recordant de **«Ús dels metapaquets»** que els metapaquets tenen el prefix `task-`, simplement especificar aquests paràmetres del llenguatge a l'arrencada, i després afegir els paquets de prioritat estàndard i tots els metapaquets descoberts a la nostra llista de paquets de la següent manera:

```
$ mkdir live-gnome-ch
$ cd live-gnome-ch
$ lb config \
  -a i386 \
  -k 486 \
```

```
--bootappend-live "boot=live components locales=de_CH.UTF-8 keyboard-↵  
    layouts=ch" \  
--debian-installer live  
$ echo '! Packages Priority standard' > config/package-lists/standard.list.↵  
chroot  
$ echo task-gnome-desktop task-german task-german-desktop >> config/package↵  
-lists/desktop.list.chroot  
$ echo debian-installer-launcher >> config/package-lists/installer.list.↵  
chroot
```

913

Tenir en compte que s'ha inclòs el paquet *debian-installer-launcher* per a llançar l'instal·lador des de l'escriptori en viu, i també s'especifica el nucli 486, ja que actualment és necessari que l'instal·lador i el nucli del sistema viu coincideixin perquè el llançador funcioni correctament.

## Apèndix

## Style guide

## 19. Guia d'estil

### 19.1 Instruccions per als autors

Aquesta secció s'ocupa d'algunes consideracions generals a tenir en compte al escriure documentació tècnica per a *live-manual*. Es divideixen en aspectes lingüístics i procediments recomanats.

**Nota:** Els autors han de llegir primer **<Contribuir a aquest document>**

#### 19.1.1 Característiques lingüístiques

- *Utilitzar un anglès planer*

Tenir en compte que un alt percentatge dels lectors no són parlants nadius d'anglès. Així que, com a regla general, intentar utilitzar frases curtes i significatives, seguides d'un punt i a part.

Això no vol dir que s'hagi d'utilitzar un estil simplista i ingenu. És un suggeriment per intentar evitar, en la mesura del possible, les oracions subordinades complexes que fan que el text sigui difícil d'entendre per als parlants no nadius d'anglès.

- *Varietat d'anglès*

Les varietats d'anglès més esteses són el britànic i l'americà, així que és molt probable que la majoria dels autors acabin utilitzant l'una o l'altra. En un entorn de col·laboració, la varietat ideal seria "l'anglès internacional", però és molt difícil, per no dir impossible, decidir quina varietat d'entre totes les existents, és la millor.

Esperem que les diferents varietats es puguin barrejar sense crear malentesos, però en termes generals s'ha d'intentar ser coherent i

abans de decidir sobre l'ús de l'anglès britànic, l'anglès americà o qualsevol altra varietat, fer una ullada a com escriuen altres persones i tractar d'imitar l'estil.

- *Ser equilibrat*

S'ha de ser imparcial. Evitar incloure referències a ideologies totalment alienes a *live-manual*. L'escriptura tècnica ha de ser el més neutral possible. Està en la naturalesa mateixa de l'escriptura científica.

- *Ser políticament correcte*

Evitar el llenguatge sexista tant com sigui possible. Si es necessita fer referència a la tercera persona del singular utilitzar preferiblement "they" en lloc de "he" or "she" o invents estranys com per exemple "s/he" o "s(he)".

- *Ser concís*

Anar directament al gra i no fent voltes. Donar tota la informació necessària, però no afegir més informació de la necessària, és a dir, no explicar detalls innecessaris. Els lectors són intel·ligents. Es presumeix algun coneixement previ per la seva part.

- *Minimitzar la feina de traducció*

Tenir en compte que qualsevol cosa que s'escrigui haurà de ser traduïda a diverses llengües. Això implica que un nombre de persones hauran de fer un treball extra si s'agrega informació innecessària o redundant.

- *Ser coherent*

Com s'ha suggerit abans, és gairebé impossible estandarditzar un document escrit en col·laboració en un tot perfectament unificat. No obstant això, s'aprecia tot esforç per escriure d'una manera coherent amb la resta dels autors.

- *Cohesió*



938 Utilitzar connectors del discurs perquè el text sigui coherent i sense  
ambigüitats. (Normalment s'anomenen connectors).

939 • *Ser descriptiu*

940 És preferible descriure l'assumpte en un o diversos paràgrafs en  
lloc d'utilitzar una sèrie de oracions en un típic estil de "changelog".  
Cal descriure-ho! Els Lectors ho agrairan.

941 • *Diccionari*

942 Buscar el significat de les paraules en un diccionari o una  
enciclopèdia si no es sap com expressar certs conceptes en  
anglès. Però cal tenir en compte que un diccionari pot ser el  
millor amic o pot convertir-se en el pitjor enemic si no es sap com  
utilitzar-lo correctament.

943 L'anglès té el vocabulari més gran que existeix (amb més d'un milió  
de paraules). Moltes d'aquestes paraules són préstecs d'altres  
llengües. Al buscar el significat de les paraules en un diccionari  
bilingüe la tendència d'un parlant no nadiu d'anglès és triar la que  
sona més semblant en la seva llengua materna. Sovint, això es  
converteix en un discurs excessivament formal que no sona ben  
natural en anglès.

944 Com a regla general, si un concepte es pot expressar utilitzant  
diferents sinònims, és un bon consell triar la primera paraula  
proposada pel diccionari. En cas de dubte, és sovint correcte  
elegir les paraules d'origen germànic (Normalment paraules  
monosíl·labes). Tenir en compte que aquestes dues tècniques  
poden produir un discurs més aviat informal, però almenys la  
elecció de paraules serà d'ampli ús i acceptació general.

945 L'ús d'un diccionari de frases fetes es recomanable. Són molt  
útils quan es tracta de saber quines paraules solen aparèixer  
junt.

946 Com s'ha dit abans, és una bona pràctica aprendre del treball dels

altres. L'ús d'un motor de recerca per comprovar com altres autors  
utilitzen certes expressions pot ajudar molt.

• *Falsos amics, modismes i altres expressions idiomàtiques*

Compte amb els falsos amics. No importa com de competent un  
és en un idioma estranger, de tant en tant es pot caure en el  
parany dels anomenats "falsos amics", paraules que s'assemblen  
en dos idiomes però els significats o usos poden ser completament  
diferents.

Intentar evitar els modismes tant com sigui possible. Els  
"modismes" són expressions que tenen un significat completament  
diferent del que les seves paraules per separat semblen voler dir.  
De vegades, els modismes poden ser difícils d'entendre fins i tot  
per als parlants nadius d'anglès!

• *Evitar l'argot, les abreviatures, les contraccions...*

Tot i que s'anima a utilitzar un anglès senzill i planer, l'escriptura  
tècnica pertany al registre formal de la llengua.

Intentar evitar l'argot, les abreviatures inusuals que són difícils  
d'entendre i per sobre de tot, les contraccions que tracten d'imitar  
el llenguatge parlat. Per no parlar d'expressions familiars o típiques  
de l'irc.

### 19.1.2 Procediments

• *Provar abans d'escriure*

És important que els autors provin els seus exemples abans  
d'afegir-los a *live-manual* per assegurar-se que tot funciona com  
es descriu. Fer les proves en un entorn chroot net o en una  
màquina virtual pot ser un bon punt de partida. A més, seria ideal  
si les proves fossin dutes a terme en diferents ordinadors amb  
un maquinari diferent per detectar els possibles problemes que  
puguin sorgir.

956	• <i>Exemples</i>	
957	Quan es posa un exemple mirant de ser el més específic possible. Un exemple és, després de tot, només un exemple.	
958	Sovint és millor utilitzar una línia que només s'aplica a un cas concret que l'ús d'abstraccions que poden confondre als lectors. En aquest cas es pot donar una breu explicació dels efectes de l'exemple proposat.	
959	Hi pot haver algunes excepcions quan l'exemple suggereixi l'ús d'algunes ordres potencialment perilloses que, si s'utilitzen incorrectament, poden provocar la pèrdua de dades o altres efectes indesitjables similars. En aquest cas, s'haurà de proporcionar una explicació detallada sobre els possibles efectes secundaris.	
960	• <i>Enllaços externs</i>	
961	Els enllaços a llocs externs només s'han d'utilitzar quan la informació en aquests llocs és crucial per a comprendre un punt especial. Tot i això, intentar utilitzar els enllaços a llocs externs el menys possible. Els enllaços d'internet poden canviar de tant en tant, donant lloc a enllaços trencats i deixant els arguments en un estat incomplet.	
962	A més, la gent que llegeix el manual sense connexió no podrà seguir els enllaços.	
963	• <i>Evitar les marques i les coses que violen la llicència sota la qual es publica el manual</i>	
964	Intentar evitar les marques tant com sigui possible. Tenir en compte que altres projectes derivats poden fer ús de la documentació que escrivim. Així que estem complicant les coses per a ells si s'afegix determinat material específic.	
965	<i>live-manual</i> es publica sota la llicència GNU GPL. Això té una sèrie d'implícacions que s'apliquen a la distribució dels materials (de	
	qualsevol tipus, incloent-hi gràfics o logotips amb drets d'autor) que es publica amb ell.	
	• <i>Escriure un primer esborrany, revisar, editar, millorar, fer de nou si és necessari</i>	966
	- Pluja d'idees!. Es necessari organitzar les idees primer en una seqüència lògica d'esdeveniments.	967
	- Quan d'alguna manera ja s'han organitzat aquestes idees en la ment, escriure un primer esborrany.	968
	- Revisar la gramàtica, la sintaxi i l'ortografia. Tenir en compte que els noms propis de les versions, com ara <b>jessie</b> o <b>sid</b> , no s'han d'escriure en majúscula quan es refereixen als noms en clau. Per tal de comprovar l'ortografia es pot executar el target "spell". És a dir, make spell	969
	- Millorar les frases i refer qualsevol part si és necessari.	970
	• <i>Capítols</i>	971
	Utilitzar el sistema de numeració convencional dels capítols i subtítols. Per exemple 1, 1.1, 1.1.1, 1.1.2 ... 1.2, 1.2.1, 1.2.2 ... 2, 2.1 ... i així successivament. Veure marcat a continuació.	972
	Si s'ha d'enumerar una sèrie de passos o etapes en la descripció, també es poden utilitzar els nombres ordinals: primer, segon, tercer ... o en primer lloc, llavors, després, per fi ... Alternativament, es poden utilitzar punts.	973
	• <i>Marcat</i>	974
	I per últim però no menys important, <i>live-manual</i> utilitza <SiSU> per processar els fitxers de text i produir múltiples formats. Es recomana fer una ullada al <manual de SiSU> per a familiaritzar-se amb el seu marcat, o bé escriure:	975
		976

```
$ sisu --help markup
```

Aquests són alguns exemples de marcat que poden ser útils:

- Per a l'èmfasi/negreta:

```
*{foo}* o !{foo}!
```

produeixen: **foo** o **foo** . S'usen per emfatitzar certes paraules clau.

- Per a la cursiva:

```
/ {foo} /
```

produeix: *foo*. S'usa, per exemple, per als noms dels paquets Debian.

- Per a monospace:

```
# {foo} #
```

produeix: `foo`. S'usa per exemple, per als noms de les ordres. I també per ressaltar algunes paraules clau o coses com les rutes.

- Per a blocs de codi:

```
code{
```

```
$ foo
# bar

}code
```

produeix:

```
$ foo
# bar
```

S'utilitza `code{` per a obrir i `}code` per a tancar els blocs. És important recordar deixar un espai al principi de cada línia de codi.

## 19.2 Directrius per als traductors

Aquesta secció s'ocupa d'algunes consideracions generals a tenir en compte a l'hora de traduir el contingut de *live-manual*.

Com a recomanació general, els traductors haurien d'haver llegit i entès les regles de traducció que s'apliquen als seus llenguatges específics. En general, els grups de traductors i les llistes de correu proporcionen informació sobre com produir traduccions que compleixin amb els estàndards de qualitat de Debian.

**Nota:** Els traductors també han de llegir [«Contribuir a aquest document»](#). En particular, la secció [«Traducció»](#).

### 19.2.1 Consells de traducció

- *Comentaris*

El paper del traductor és transmetre el més fidelment possible el

significat de les paraules, oracions, paràgrafs i textos de com van ser escrits pels autors originals al seu idioma.

999 Per tant, aquests, s'han d'abstenir d'afegir comentaris personals o informacions addicionals pel seu compte. Si es vol afegir un comentari per als traductors que treballen en els mateixos documents, es poden deixar a l'espai reservat per a això. És a dir, la capçalera de les cadenes dels fitxers **po** precedits pel signe # . La majoria dels programes gràfics de traducció poden manejar aquest tipus de comentaris automàticament.

1000 • *NT, Nota del Traductor*

1001 És perfectament acceptable però, incloure una paraula o una expressió entre parèntesi en el text traduït si, i només si, aixó fa que el significat d'una paraula o expressió difícil sigui més clara per al lector. Dins dels parèntesis, el traductor ha de posar de manifest que l'addició és seva utilitzant l'abreviatura "NT" o "Nota del traductor".

1002 • *Frases impersonals*

1003 Els documents escrits en anglès fan un gran ús de la forma impersonal "you". En alguns altres idiomes que no comparteixen aquesta característica, això pot donar la falsa impressió que els textos originals s'adresen directament al lector quan en realitat n'ho fan. Els traductors han de ser conscients d'aquest fet i ho han de reflectir en el seu idioma amb la major precisió possible.

1004 • *Falsos amics*

1005 El perill dels "falsos amics" explicat anteriorment s'aplica especialment als traductors. Tornar a comprovar el significat dels falsos amics sospitosos en cas de dubte.

1006 • *Marcats*

1007 Els traductors que treballin inicialment amb els fitxers **pot** i posteriorment amb els fitxers **po** trobaràn moltes característiques

de marcat en les cadenes. Es pot traduir el text, sempre que sigui traduïble, però és extremadament important que s'utilitzi exactament el mateix marcat que a la versió original en anglès.

• *Blocs de codi*

1008

Tot i que els blocs de codi són generalment intraduïbles, incloure'ls en la traducció és l'única manera de conseguir una traducció completa al 100%. I encara que això signifiqui més feina al principi, ja que pot requerir la intervenció dels traductors s'hi ha canvis en el codi, és la millor manera, a la llarga, per identificar el que s'ha traduït i el que no al comprovar la integritat dels fitxers .po.

1009

• *Salts de línia*

1010

Els texts traduïts han de tenir exactament els mateixos salts de línia que els texts originals. Aneu amb compte de pressionar la tecla "Enter" o escriure si apareix als fitxers originals. Aquests salts de línies apareixen sovint, per exemple, en els blocs de codi.

1011

No confondre's, això no vol dir que el text traduït hagi de tenir la mateixa longitud que la versió en anglès. Aixó és gairebé impossible.

1012

• *Cadenes intraduïbles*

1013

Els traductors no han de traduir mai:

1014

- Els noms en clau de les versions (que han de ser escrits en minúscules)

1015

- Els noms dels programes

1016

- Les ordres que es posen com a exemples

1017

- Les metadades (apareixen sovint entre dos punts **:metadata:** )

1018

- Els enllaços

1020

- Les rutes

1019

## SiSU Metadata, document information

**Title:** Manual de Live Systems

**Creator:** Projecte Live Systems <debian-live@lists.debian.org>

**Rights:** Copyright: Copyright (C) 2006-2014 Live Systems Project

License: Aquest programa és un programari lliure: es pot redistribuir i/o modificar sota els termes de la Llicència Pública General de la GNU com és publicada per la Free Software Foundation, ja sigui la versió 3 de la Llicència, o (si ho preferiu) qualsevol versió posterior.

Aquest programa es distribueix amb l'esperança que sigui útil, però sense cap garantia, fins i tot sense la garantia implícita de COMERCIALITZACIÓ o ADEQUACIÓ PER A PROPÒSITS DETERMINATS. Vegeu la Llicència General Pública de la GNU per a més detalls.

Haurieu de rebre una còpia de la Llicència Pública General de la GNU amb aquest programa. Si no és així, consulteu <<http://www.gnu.org/licenses/>>.

El text complet de la Llicència Pública General de la GNU es pot trobar a /usr/share/common-licenses/GPL-3.

**Publisher:** Projecte Live Systems <debian-live@lists.debian.org>

**Date:** 2014-10-25

### Version Information

**Sourcefile:** live-manual.ssm.sst

**Filetype:** SiSU text 2.0, UTF-8 Unicode text, with very long lines

**Source Digest:** SHA256(live-manual.ssm.sst)=1ac263cb3382657c94645b52-30928de3de84d02cc8c9e8f6940d738252a66370

### Generated

**Document (ao) last generated:** 2014-10-25 13:11:31 +0000

**Generated by:** SiSU 5.7.1 of 2014w41/7 (2014-10-19)

**Ruby version:** ruby 2.1.3p242 (2014-09-19) [x86\_64-linux-gnu]